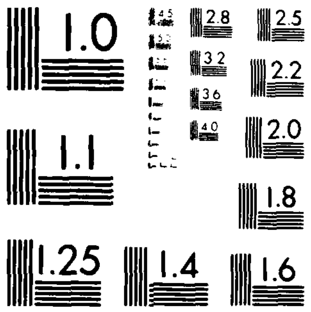
AD-A086 571 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
THE UTILIZATION OF REQUIREMENT STATEMENT METHODOLOGIES IN THE U--ETC(U)
MAR 80 F A PETRIE
UNCLASSIFIED NL

1 OF 2
AD
A086571



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

LEVEL

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

ADA086571

THESIS

(6) The Utilization of Requirement Statement Methodologies in the United States Navy and Their Impact on Systems Acquisition.

by

(11) Frederic Andrew/Petrie, III

(11) Mar[redacted]80

(12) 1:3

Thesis Advisor: N. F. Schneidewind

Approved for public release;distribution unlimited

DTIC
ELECTE
S JUL 1 5 1980 D
A

DDC FILE COPY

251 80 7 14 092

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A086 571	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Utilization of Requirement Statement Methodologies in the United States Navy and Their Impact on Systems Acquisition		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 80
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Frederic Andrew Petrie III		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 80
		13. NUMBER OF PAGES 122
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

requirements specifications
SREM
PSL/PSA
systems acquisition

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The success or failure of weapons systems software projects can often be traced back to the project's definition phase. There currently exists in the literature many articles dealing with the problems inherent in the development of requirements specifications. This thesis reviews some of the problems and examines an evolving, disciplined method to better state the user's requirements, called Requirement Statement Languages (RSL)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601

unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Two automated systems utilizing RSL, SREM and PSL/PSA, are reviewed as to their strengths and weaknesses in system definition and development, particularly as they are currently used in the Navy. Also discussed are how these systems may be utilized in the Navy's system acquisition process and recommendations are made as to how the Navy can incorporate such software technology.

Accession For
NTIS GRA&I [✓]
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

Approved for public release; distribution unlimited

The Utilization of Requirement Statement Methodologies in the United States Navy and Their Impact on Systems Acquisition

by

Frederic Andrew Petrie III
Lieutenant, United States Navy
B.S., United States Naval Academy, 1974

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March, 1980

Author Frederic A. Petrie III

Approved by: Norman F. Schneidewind
Thesis Advisor

[illegible]
Second Reader

[illegible]
Chairman, Department of Computer Science

[illegible]
Dean of Information and Policy Sciences

ABSTRACT

The success or failure of weapons systems software projects can often be traced back to the project's definition phase. There currently exists in the literature many articles dealing with the problems inherent in the development of requirements specifications. This thesis reviews some of the problems and examines an evolving, disciplined method to better state the users' requirements, called Requirement Statement Languages (RSL). Two automated systems utilizing RSL, SREM and PSL/PSA, are reviewed as to their strengths and weaknesses in system definition and development, particularly as they are currently used in the Navy. Also discussed are how these systems may be utilized in the Navy's system acquisition process and recommendations are made as to how the Navy can incorporate such software technology.

TABLE OF CONTENTS

I. INTRODUCTION

The Department of Defense is faced every year with the development of clear, concise requirements specifications for hundreds of systems. These specifications serve as a vehicle for the development of systems that are vital to the support of the missions of the armed services and, more importantly, the overall national defense. It is with these specifications that the relative success or failure of the development of a system rests.

For example, take a real-time combat system. The first feature of such a system is that it is required to be highly reliable ; ideally it should function properly at all times. This system is required to be extremely flexible in response. Conditions, be they meteorological, electromagnetic, tactical, etc., can vary rapidly thus forcing the development of a system that is almost self-adjusting. Automation is a prerequisite for such a system due to the short length of engagements forseen in future military encounters and because the programs that drive such systems are large and quite complex, receiving many inputs and then performing multiple command and control functions. [1] [2]

Unfortunately, the above requirements cannot be adequately tested in anything other than either an operational evironment [1] or a highly realistic simulation, making development of precise requirements specifications even more critical.

Much to the consternation of the project offices and end-users, many problems exist in the area of requirements specifications. These problems cost the taxpayers millions of dollars and create countless headaches for the military as it is delivered systems that do not perform as expected.

What are some of the causes? Part of it is due to the fact that some of the projects are simply too ambitious. All of the requirements that are forced on a system raise the level of complexity to the point where the project is absolutely infeasible in terms of technology, time, and money. On a higher level, too often ambiguous, incomplete, and untestable requirements, symptoms of poor communication, are forced upon contractors, often coupled with documentary information that is factually incorrect. [3]

Once the requirements specifications reach the contractor there immediately is the likelihood of misinterpretation due to the aforementioned ambiguity and incompleteness, plus the fact that many of the requirements are highly conceptual in nature. The programmer who views structured design as a handicap and is more concerned with code than with overall design further exacerbates the problem [2].

Between the initial misinterpretation and the programmer's code comes the problem with the design phase. Too often there seems to be subsystem optimization at the expense of the overall system. [2] This is the result of the pressures of time and "pride of workmanship" rather than an attempt to undermine. Development and maintenance of structure charts are another problem. [2] Typical charts measure 20′ by 10′ and take days to update. Finally, inconsistent and ill defined approaches in this phase have resulted not only in lackluster results, but also in poor presentation, inconsistency, incompleteness, and general confusion about the status of the project in question. [3]

Hammond et al. [4] noted the importance of careful design; that errors originating during the design phase are very costly to correct during later development. They cited a DOD report that estimated that design errors discovered during the operation of a system cost 8-9 times more to correct than those detected during the detailed design. Munson [5] cited another DOD report that stated that approximately 60-70% of its software dollars ($2 billion in 1976) are spent after software has been tested and delivered.

It is the intent of this thesis to look at the problem of requirements specifications in terms of what they are, how they should be properly utilized, and how their effectiveness can be enhanced when developed through the relatively new concepts of requirement statement languages and software requirements methodologies. Two of the more mature systems utilizing this concept, the Software Requirements Engineering Methodology and the Problem

Statement Language/Problem Statement Analyzer, will be reviewed as to their capabilities and possible limitations in development of requirements specifications in DOD. This thesis will also examine how these systems may be utilized in the Navy's system acquisition process and will make recommendations as to how the Navy can incorporate such software technology.

II. BACKGROUND

Mullery [4], Balzer and Goldman [6], and Heninger [7] have all addressed the question of exactly what the overall aims of requirements specifications should be and how these aims can be realized.

The most basic aim of a requirement specification should be that of defining the requirement so that the system may be implemented later and be proven to have been implemented correctly and also to define the requirement so that the customer and end-user can verify that the system will perform the requisite functions. [4] This forces the issue of clarity and the elimination of ambiguity. Forethought, systematic development of specifications, and error checking of system logic on a very high level are paramount.

The requirement specification should take a modular approach to the task of system definition. The specification must be localized and loosely coupled [6] and should specify external behavior only so as not to force a particular solution [7]. Since during system development many modifications are likely, the separation of particular requirements (localizing and loosely coupling) contributes greatly to the overall flexibility of the system development and minimizes the side effects of modifications. To carry this thought even further, the specifications must be tolerant of any omissions and permit augmentation of requirements at some future point [6] [7]. This would seem to defeat the purpose of structured formulation of requirements specifications but it is necessary due to the highly iterative nature of large system design and the uncertainties of the human thought process.

As a design tool, these specifications should be consistent and compatible for each of the individual requirements [3]. Such things as naming conventions for the various components and interfaces between modules must be considered. Also, avoidance of unnecessary repetition of information so as to reduce bulk and prevent possible confusion is important.

Three other key aids to design are (1) to define each module so that all parties involved in the design of the system can grasp the overall concept of the system [5], (2) to characterize acceptable responses to undesired events, and (3) specify constraints, particularly in the area of hardware interfaces [7]. All of these serve as a means of defining the overall system and its purpose.

Heninger went even further by stating that the specifications should serve as a reference tool, having the ability to answer specific questions quickly, and also record forethought about system lifecycle costs. What types of changes are likely to occur? What functions would maintenance like to be able to remove easily? [7]

Merten and Teichrow [8] cited a study by the Office of Management and the Budget. The study, conducted to improve the effectiveness of systems analysts and programmers, stated that the most important way to improve the effectiveness of these personnel is to reduce the time spent on and greatly improve the efficiency of systems analysis, design, implemetations, and maintenance. Granted, this statement in and of itself says nothing new, but it does reinforce the idea of a need for a more rigorous, disciplined approach to systems design and implementation. This approach to be successful and effective must start with the requirements specifications.

Willis and Jensen [2] noted the shortcomings of so-called "methodologies" vis a vis engineering when they described methodologies as being generic and subject to interpretation. Conversely, they cited engineering as a discipline that stresses standardization and serves as a much more effective and efficient vehicle for developing systems and conveying information and concepts. They went even further by explaining that the fundamental precepts of systems engineering must be preciseness, consistency, and completeness of applications. They also felt the use of automated tools to be necessary for training, configuration control, and quality control.

Since computers are used for design, modeling, and simulation in other areas, why not use them to generate requirements and overall system design?

III. REQUIREMENTS STATEMENT LANGUAGE

On a macro level, the use of engineering principles and automated tools looks like a boon to mankind. However, whether one communicates with a computer or with a team of designers, the fact still remains that a medium is still necessary to effectively convey system requirements. For years proponents of natural languages such as English have claimed far and wide that these languages are "very high" level and that their use constitutes the wave of the future. Exception to this is taken by Jones [9]. His own independent survey noted that English is actually a very low level language as it requires 3 to 11 times as many English words to specify a program as it takes lines of assembler to code it. He found that with programs that exceed 128K lines of assembler specifications become too bulky and cease to be useful. It is at this point that "verbal communication" becomes the dominating factor.

Combining the findings of Jones with one's own experience with the vagueness and ambiguity inherent in the use of the English language, it becomes readily apparent that what is required is a requirements statement language, a language that precisely, concisely, and completely conveys to all concerned the actual user requirements.

Whereas a programming language serves as a means of communication between a programmer and a compiler or assembler, a requirements statement language (RSL) should serve as a means of communication between the user and an analyst or system designer [8]. Teichrow [10] listed three main functions of an RSL:

1. RSL should accomodate the statement of requirements of the kind that are occuring now as well as those in the future.

The future will produce hardware improvements in both quality and reliability. Parallel processing and concurrency will become more common. There will be a marked increase in the interrelationship of requirements. As the number and types of users increase, additional problems of interfacing will arise. Far greater demands for system performance and real-time applications will occur. There will also be an additional requirement for system monitoring. All of these problems and more must be taken into consideration in the design of an RSL.

2. RSL should be suitable for use by humans for determining and stating requirements.

The RSL should be so structured that it can be used by personnel on all levels, in all phases of design. This hopefully will reduce the strict dependency on the analyst as a go-between for management and design. The RSL should also be suitable for use in top-down design and should be computer testable for completeness and consistency. All of this should augment the capabilities of those involved in the defintiion of requirements.

3. RSL should be suitable for building the system to accomplish the stated requirements.

This will occur if the RSL is allowed to generate statements of requirements and not statements of data processing -- what the system is supposed to do, not how to do it. This will aid in keeping the requirements hardware independent, thus saving possible reconversion costs.

Merten and Teichrow [8] amplified the last few statements when they noted that the major purpose of the RSL is to force the user to state his requirements in a manner which does not force a particular processing procedure. However, they also noted that this is a difficult concept to impose given the techniques that are ingrained in the specification process. If followed rigorously, this should reduce the existence of illogical requirements due to poor specifications.

The concept of RSL is not new, being first developed as early as 1958, but, until recently, has not been in wide use due to the lack of ability to analyze problem definitions in the RSL, so it has been mainly relegated to use as a documentation tool [8]. Jones [9] noted that there are about 150 design languages that have been developed.

The following chapter will address a current methodology that employs a requirements statement language, namely the Software Requirements Engineering Methodology, developed by TRW for the Ballistic Missile Defense system. As noted above, there are some 150 languages; however, SREM was chosen for further discussion due to its relative maturity of development, the fact that it was developed for a major project, and because its has proven successful to some degree. Its discussion will center around its structure and its approach to system specification and design.

IV. SOFTWARE REQUIREMENTS ENGINEERING METHODOLOGY

A. BACKGROUND

In 1974 the Ballistic Missile Defense Advanced Technology Center (BMDATC) initiated sponsorship of an integrated software development research program aimed at improving the techniques for developing correct, reliable software for the proposed Ballistic Missile Defense (BMD) system. The overall program sought to cover a broad spectrum, from development of software specifications to the completion and testing of the software process design [11]. Other areas of research involved software reliability, static and dynamic validation techniques, and adaptive control and learning.

At the center of this program was the Software Requirements Engineering Program (SREP), an effort concerned with a systematic approach to the development of complete and validated software requirements. Its overall objectives were to:

1. Ensure a well defined technique for the decomposition of system requirements into structured software requirements.

2. Provide a vehicle to enable management to clearly see and understand all phases of the requirements development.

3. Ensure that requirements development was completely machine and design independent.

4. Provide for easy response to changes in systems requirements.

5. Produce testable and easily validated software requirements [11].

The product of the above program is the Software Requirements Engineering Methodology (SREM). SREM includes techniques and procedures for requirement decomposition and for managing the requirements development process [12]. Within this methodology are software support tools which were implemented to automate many of the manual activities associated with requirements engineering. Among these tools are the Requirements Statement Language (RSL), a machine-processable language for stating requirements, and the Requirements Engineering and Validation System (REVS) which supports the development of requirements written in RSL. SREM represents a different approach and philosophy for software requirements engineering. It utilizes a flow orientation that precludes many of the problems inherent in the classical functional hierarchy.

The functional hierarchy (Figure 1.) is the most prevalent way to organize software requirements. In Figure 1 the boxes marked B,C, and D represent major functions of software such as tracking, guidance, etc. These major functions are broken into subfunctions down to seven to ten levels. It is from these lower levels that the requirements are written.

The first problem encountered with this approach is the requirements are written at too low a level. Though each individual subfunction can be tested for correctness, there is extreme difficulty in testing the system as a whole, i.e. top-down, against the system specification. The requirements must be developed so that each condition that could possibly be encountered can be traced down through each appropriate subfunction until the output is determined.

Another problem encountered from developing requirements at too low a level is that performance requirements are not easily derived. This is due to the fact that the timeline and accuracy budgets have to be partitioned among too many levels.

Finally, it is difficult to check for completeness and consistency. Since there is no algorithm to guide the derivation of the tree structure, there is no algorithm with provable validity to guide the analysis.

The methodology expressed in SREM encompasses four major areas of engineering activity that commence with the input of information that defines the system level requirements on

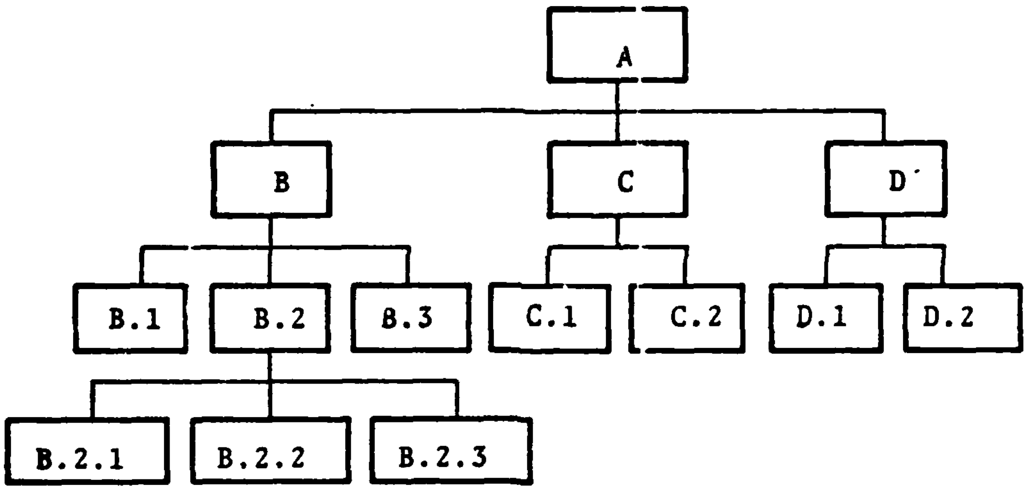


Figure 1.
Hierarchy of Functions

the Data Processing Subsystem. This information is denoted as the Data Processing System Performance Requirements (DPSPR) Specification [11]. The DPSPR includes system interface and performance requirements specifications. These enable the requirements engineer to involve himself in:

1. identification, definition, and development of the functional requirements.

2. identification, definition, and development of the performance requirements.

3. development of the Process Performance Requirements Specifications.

4. development of the analytic feasibility demonstrations.

B. SREM OBJECTIVES

The key concept in the development of SREM was that design-free functional software requirements should specify the required processing in terms of all possible responses (and the conditions for each type of response) to each input message across each interface. These functional requirements identify the required stimulus and response which are expressible in terms of Requirement Networks (R-Nets) of processing steps. Each step is defined in terms of input data, output data, and the transformations which are associated with the step [13].

Though designed for the BMD system, SREM was aimed towards any major system with the following characteristics:

1. Systems with more than 100K lines of code.

2. Time responses are critical. This is the criteria that defines a "real-time" system, i.e., receiving input, processing the information, and producing output that will in some way influence the immediate environment.

3. Processing is very intensive. A real-time system could perhaps be tasked with tracking several hundred targets.

4. Database is large but not massive. The database must be indigenous to the system; time cannot be wasted in information retrieval.

5. Technology of the object system initially is not fully understood. Justification and feasibility of the system and its possible subsystems are still an issue [11].

SREM was also designed to encompass a wide range of system development environments, ranging from systems which must deal with hard performance requirements, firm threat definition, and maximum design freedom, to systems with minimum performance requirements, flexible threat definition, and reduced design freedom.

SREM was never intended to be the ultimate panacea for the woes of system design and development. A thorough knowledge of systems engineering and data processing technology are still paramount. The utilization of SREM commences only after system analysis has identified the functions and stress points of the system; the interfaces between subsystems; top-level weapon system functions and operating rules; and the top-level weapon system functions have been allocated to the data processor.

The termination point is reached when all system requirements have been decomposed to the point where software development expertise is necessary to continue; interfaces have been defined on the element level; all responses to system stimuli have been determined; and the processing necessary to generate all required output interface messages has been identified [11].

C. SREM EVOLUTION

During the initial definition of SREM it was necessary to determine those properties required of both a specification and of the individual requirements of which it is composed. The initial considerations were that, first, a specification is a set of all requirements which must be satisfied together with the identification of the subsets which must be met concurrently. Secondly, a specification must be consistent with the laws of logic and nature before they can be realizable and legally binding. Lastly, a specification must be so stated that any delivery satisfies the specification and the user's needs.

The above considerations were further evaluated and meshed with technical, economic, and management points of view, producing several properties that were felt to be mandatory to the success of SREM. The properties that evolved include:

1. internal consistency
2. consistency with the physical universe
3. freedom from ambiguity
4. clarity
5. minimality
6. predictability of specification development
7. controllability of software development [11].

To ensure the property of freedom from ambiguity, it was mandatory that a rigorous machine-readable language be developed. By employing an unambiguous language which is translated and analyzed by a program intolerant of ambiguity, a precise statement of requirements was ensured.

Analysis of the requirements statements, through use of static and dynamic decomposition of the individual statements and analysis of the composite flow of data and processing, provides an internal consistency check. Physical universe consistency is ensured by converting the specification into a model which is tested against a model of the real world. These checks help to validate the software specification before it is imposed.

The use of selective documentation and analysis of the software specifications, when coupled with sound engineering and management techniques, provides predictability in the specification process and aids in avoiding overspecification.

D. OVERVIEW OF REVS COMPONENTS

The Requirements Engineering Validation System (REVS) is composed of three major components (Figure 2.):

1. Requirements Statement Language (RSL) translator

2. Abstract System Semantic Model (ASSM), a centralized database.

3. A set of automated tools for processing the information held by the ASSM.

The entire system is based on the ASSM, a relational database similar in concept to that used by the PSL/PSA system developed at the University of Michigan. Though the

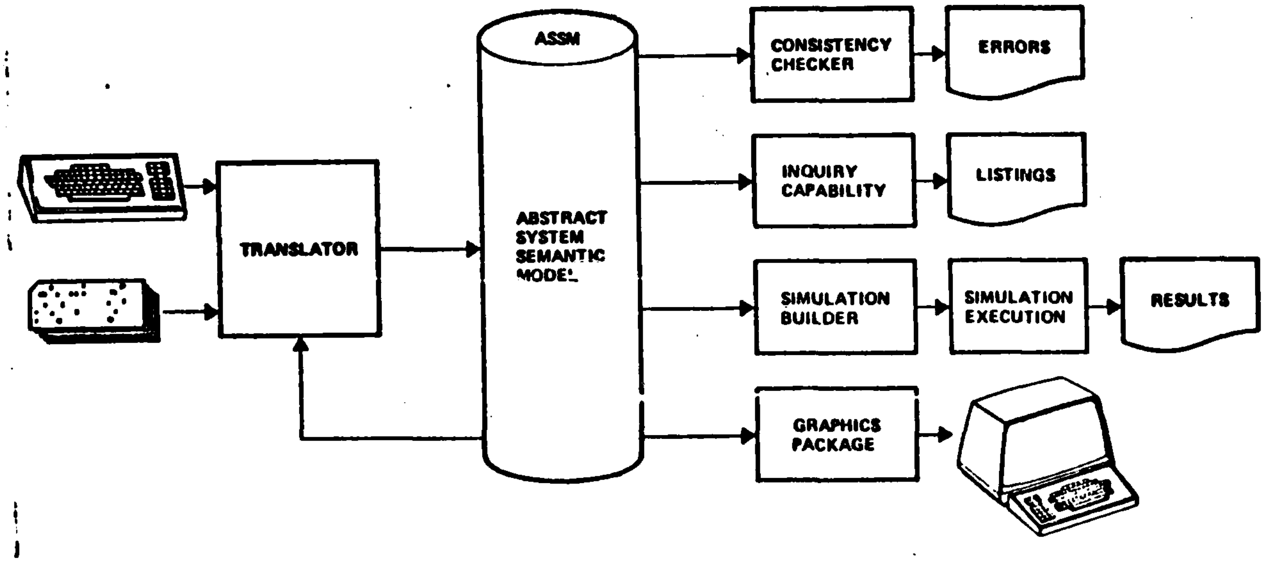


Figure 2.
SREM System

concepts are similar, the implementations differ due to the need for extensibility, configuration management, and for a flow approach for simulation being strongly stressed in the development of SREM.

The ASSM is the interface between the Requirements Statement Language and the set of automated analysis tools. This allows the extension of the language without having to take into consideration such things as operating system impact and control of the automated tools. It also allowed the RSL to be developed as a natural method in which to express requirements; not being constrained by control languages or configuration management [12].

Besides providing a means to naturally express requirements, the RSL also provides a rigorous structure that allows it to be machine-interpretable. This is due to the fact that it was designed around the specification of flow graphs of required processing steps [11, 12, 13]. These flow graphs are expressed as "structures", the product of mapping a two dimensional graph (Figure 3.) onto a one dimensional input stream (Figure 4.) [12]. The aforementioned extensibility allows the modification of the RSL to suit particular requirements and provides a means to accomodate new, unanticipated needs for stating requirements including non-procedural statements. The RSL statements and structures, once entered, are abstracted and entered into the ASSM where they can be used by the automated system tools.

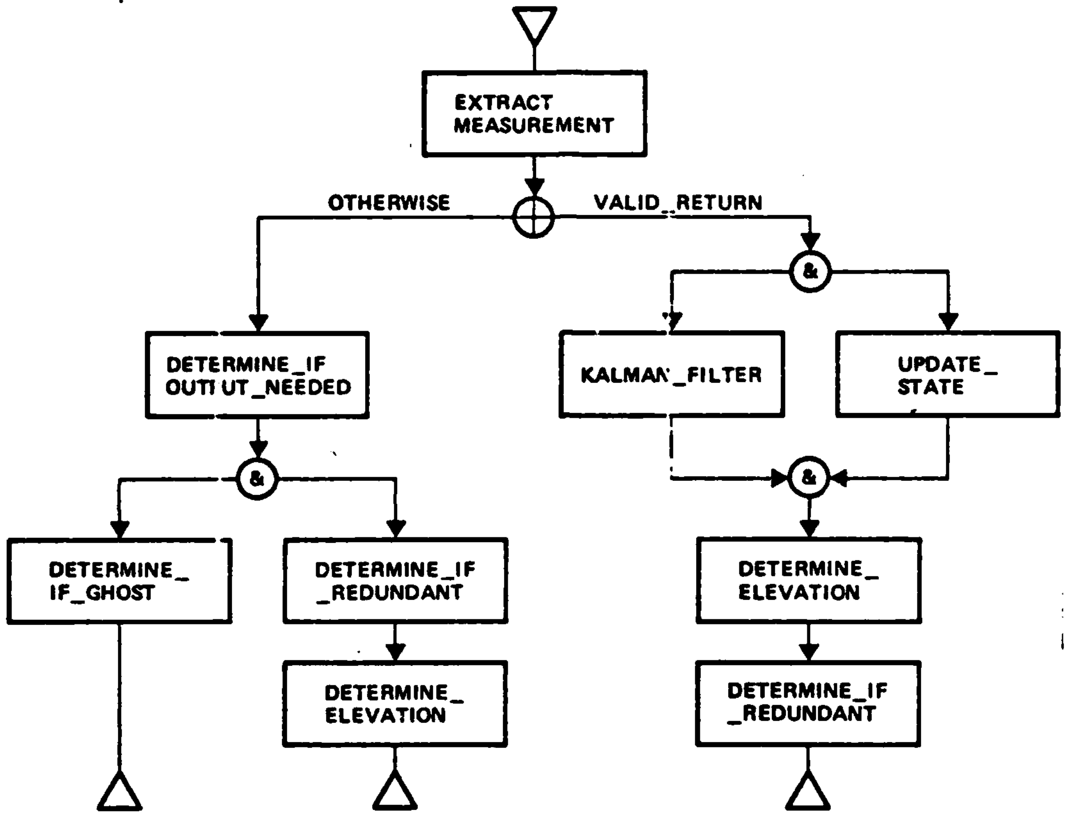


Figure 3.
R-Net

```
R_NET: PROCESS_RADAR_RETURN.
     STRUCTURE:
          EXTRACT_MEASUREMENT
          DO  (STATUS = VALID_RETURN)
                DO  UPDATE_STATE AND KALMAN_FILTER END
                DETERMINE_ELEVATION
                DETERMINE_IF_REDUNDANT
                TERMINATE
          OTHERWISE
                DETERMINE_IF_OUTPUT_NEEDED
                DO  DETERMINE_IF_REDUNDANT,
                    DETERMINE_ELEVATION,
                    TERMINATE
                AND DETERMINE_IF_GHOST,
                    TERMINATE
                END
          END
     END.
```

Figure 4.
Input Stream

The automated system tools include: interactive graphics to aid in development, specification, and modification of flow graphs; static consistency checkers used to ensure internal consistency in specifications; and an automated simulator generator and execution package which aid in dynamic testing.

These tools also ensure that portions of system specified later than some segments will be consistent since their connectivity with the early segments was defined at the highest levels. This is a particularly attractive feature as it allows system design to progress without all segments developing at the same pace and allows several persons to participate in the design process. Additionally, any extensions of the system are forced to be compatible with all prior specifications since any incompatibility would preclude entering the extension into the ASSM.

The next several sections will go into greater detail as to some of the specific mechanics employed by the aforementioned components. This information is derived from the papers by Alford et al. [11] and Bell, Bixler, and Dyer [12].

E. REQUIREMENTS STATEMENT LANGUAGE

Chapter III pointed out the findings of Jones [9]; that the use of English for documentation and specification is too often unsatisfactory due to the ambiguity inherent in the language. Alford [14] noted the inability to provide an effective means of ensuring traceability and testability of requirements; that in nearly every software project that has failed, the requirements were accused of being late, incomplete, over-constraining, or just plain wrong. In order to overcome these and other problems the first of three goals established during the initial development of SREM was to develop a language for stating requirements that addressed the properties of unambiguity, design freedom, testability, modularity and communicability [14].

The language that evolved, RSL, is an artificial language that incorporates naturalness of expression. Through use of the flow approach to defining requirements, it provides information on how pieces of the system will fit together, something not possible when the hierarchy of functions approach to specifying requirements is employed. Additionally, since the language can precisely define concepts and constrains the semantics to a simple level of detail, the risk of ambiguity is significantly reduced and only the true requirements of the system evolve.

1. Flows

The traditional hierarchy of functions approach to requirements specifications, currently mandated in DOD MIL-STD 490, describes the operations that each module is expected to perform, rendering the requirements to little more than program specifications. This method fails to adequately address the sequence of operations and the communication between modules, thus creating problems with real-time systems. In order to overcome this, RSL is structured to represent a stimulus/response approach or a "flow". Each flow is initiated by some "stimulus" or input and cascades down through the various functions, producing the appropriate response until the processing is completed. By utilizing this approach the exact sequence of processes becomes explicitly thereby enhancing testability.

The flows, commonly known a Requirements Networks or R-Nets, consist of nodes, which specify an operation, and their connecting arcs. The basic nodes consist of ALPHAs, which are the specifications of functional processing steps, and SUBNETs, which are specifications of processing flows at a lower level in the hierarchy. As noted previously, these nodes are single entry, single exit, however, more complex flows may be specified by use of structured nodes which enable the system to execute multiple flow paths. These structured nodes include AND, OR, and FOR EACH.

The AND node specifies that its eminating arcs leading to further nodes are mutually order-independent, able to be executed sequentially in any order or in parallel. The rejoining, or fan in, of the arcs at the end of the AND structure specifies a synchronization point; the execution of the processes as specified by each path in the structure must be completed in order to trigger an output from the structure.

The OR node is similar to the IF-THEN-ELSE construct in structured programming. The complete execution of all processes specified on any or all paths will trigger an output from this structure.

The FOR EACH node is similar to a loop construct in structured programming. It has only one processing path and the number of times that this path is looped through is based on the number of elements contained in a set. For example, in a tracking problem an update in the range for each target may be requested.

Because the syntactic structure of the R-Nets is similar to that of structured programming, it aids the requirements engineer in determining areas that are vague or ambiguous, in communicating with others, and in utilizing automated analysis tools.

2. Extensions

As mentioned previously, RSL incorporates the concept of extensibility so that new concepts that may develop in the future may be easily integrated into the existing system. The requirements of real-time systems is one of the primary forces behind the dynamic nature of state-of-the-art developments in digital processing and computing. Coupled with the evolutionary nature of weapons systems requirements, such as new interfacing or processing techniques, the situation would clearly render a language with fixed concepts ineffective.

By keeping the underlying architecture of RSL simple it has been possible to incorporate extensibility through use of four primitives:

a. Elements

Elements are the equivalent of nouns in English and describe the properties of each element. Elements include ALPHA, DATA (class of conceptual pieces of data), and R-NET (class of processing flow specifications).

b. Relationships

These are the equivalent of English verbs or more precisely a statement of association between two elements such as DATA INPUT TO ALPHA. It should be noted that this is a non-commutative relation; a distinct subject and object element are expressed.

c. Attributes

Attributes are similar to adjectives in English are used to formalize important properties of elements. Associated with it are a set of values which may include numbers, mnemonic names, or text strings. INITIAL VALUE and PRESENT RANGE are examples of attributes of type DATA.

d. Structures

Structures are the mapping of two-dimensional graph structures into a one-dimensional stream of computer input. They serve as a model of flows through the various processing steps.

As noted above, these four primitives define the structure of RSL. The structure in itself is not extensible; however, the primitives enable the user to define new types of elements, relationships, and attributes into the language in order to express new concepts.

Figure 5 gives an example of how ALPHA, DATA, RELATIONSHIPS, and new elements are defined.

```
ALPHA: EXTRACT_MEASUREMENT.
        INPUTS: CORRELATED_RETURN.
        OUTPUTS: VALID_RETURN, MEASUREMENT.
        DESCRIPTION: "DOES RANGE SELECTION PER
                CISS REFERENCE 2 - 7".
        ENTERED BY: "M. RICHTER".

ALPHA: DETERMINE_IF_REDUNDANT.
        INPUTS: CORRELATED_RETURN.
        OUTPUTS: REDUNDANT_IMAGE.
        DESCRIPTION: "THE IMAGE OF THE RADAR RETURN
                IS ANALYZED TO DETERMINE IF IT IS
                REDUNDANT WITH ANOTHER IMAGE".
        ENTERED BY: "F. BURNS".

DATA: MEASUREMENT.
        INCLUDES: RANGE_MARK_TIME, AMPLITUDE,
                RANGE_VARIANCE, RD_VARIANCE,
                R_AND_RD_CORRELATION.
        DESCRIPTION: "THIS IS THE ESSENCE OF THE
                INFORMATION IN THE RETURN".
        ENTERED BY: "F. BURNS".

ORIGINATING REQUIREMENT:
                DPSPR_3_2_2_A_FUNCTIONAL.
        DESCRIPTION: "ACTION: SEND RADAR ORDER
                INFORMATION: RADAR ORDER. IMAGE
                (REDUNDANT)".
        TRACES TO: ALPHA COMMAND_PULSES
                   ALPHA DETERMINE_IF_REDUNDANT
                   MESSAGE RADAR_ORDER_MESSAGE
                   DATA REDUNDANT_IMAGE
                   ENTITY IMAGE.
        ENTERED BY: "T.E. BELL".
```

Figure 5.
RSL Definitions

3. Translator

The purpose of the translator is to analyze RSL statements and make entries into the ASSM corresponding to the meaning of the statements. It accomplishes this by extracting the RSL primitives which exist in the input statements and then mapping them to constructs in the ASSM. The translator can also perform modifications and deletions from the database as commanded by RSL statements and also perform consistency check on the incoming statement to prevent duplication of element name or an illegal relationship. Additionally, it also handles the introduction of extensions with great care; the introduction may invalidate a large segment of the requirements. For this reason a lockout mechanism was designed to control the use of extensions and enforce a disciplined use of the power of RSL.

F. THE ABSTRACT SYSTEM SEMANTIC MODEL (ASSM)

The RSL statements that are entered into REVS are analyzed and their representation is entered into the ASSM, a database that maintains information about the system being designed in an abstract, relational model. Since checks are made for syntax and semantics before information is entered, it is possible to employ the various tools or REVS, assured of data format correctness. Also included in the ASSM entries are all extensions, including core concepts (basic RSL) and additions and modifications to specific projects.

They, too, are available for immediate use as soon as they are entered.

The information contained in the ASSM is not simply a string of RSL statements. Rather, it is a relational model where elements are represented by nodes and their relationships are represented as connections. Attributes and their values consist of a node for the value and a connection to the node for which the value is attributed. This representation facilitates retrieval of information, particularly in complex combinations of relationships and permits queries about specific information or relationships such as finding all DATA elements which are not INPUT TO anything.

The centralization of information in the ASSM is mandatory due to the large numbers of individuals who enter additions, deletions, and modifications to the various system requirements. This centralization ensures that all involved are working with a repository of information that is current; they can immediately see the effect of their work on other engineers, the characteristics of parts of the system that other people are defining, and the current status of their own work. In addition, centralization aids in configuration management (where blocking of modifications freezes the configuration) and in checking for consistency throughout the entire system.

G. AUTOMATED TOOLS

For large software projects, it is necessary to employ the services of many individuals to develop requirements for different segments of the system; each formulating the RSL descriptions for his/her particular part of the system. The mechanisms for imposing discipline and control on this process are the automated tools provided by REVS. These tools aid the engineer in identifying the various areas that require further development, resolve conflicts, and evaluate inputs. Since the requirements engineering process is of an iterative nature, these tools help to evaluate the entire system when various milestones are reached.

1. Interactive Graphics

The interactive graphics facility of REVS enables the engineer to input, modify, or display R-NETS. It is possible to use it in lieu of the translator for the specification of the flow portion of the requirements and it can be used to generate a graphic display of an R-NET previously entered. The two-dimensional nature of graphics serves to provide a more easily understood representation than a one-dimensional input stream; however, the facility allows the use of both graphics and the RSL language for representation of the R-NETs.

Along with the graphics are a full range of editing capabilities. A new R-NET may be constructed or one previously entered may be modified. At the end of the session the new R-NET is entered into the ASSM in place of the old one. From a menu, the user may select functions to position, connect, and delete nodes, to move them, disconnect them from other nodes, or to change their associated name and commentary. Finally, the size of an R-NET is not size-limited due to the zoom-in, zoom-out, and scroll functions.

2. Simulation

Simulation offers an effective means by which to test consistency, completeness, and validity of requirements. The building of simulations must be automatic to preclude divergence of the requirements from the simulation and to allow rapid response and analysis of change.

The automatic simulation generation in RFVS takes the ASSM representation of the requirements and generates from it simulations of the system. The System Environment and Threat Simulation (SETS) program is the driver for the software requirements model.

SETS provides all stimuli necessary for each processing option and also accepts and properly executes all valid commands. SETS is structured to simulate the required actions, calculate how long the activity would have taken in a real system, and make the results of the activity available to the software at the proper simulated time. Because of the asynchronous nature of real-time systems, R-NET timing is implicitly modeled.

SETS takes the ASSM representation of the requirements and puts them into simulation code written in

PASCAL. The flow structure of each R-NET is used to develop a PASCAL procedure whose control implements that of the R-NET structure. Each processing step (ALPHA) on the R-NET becomes a call to procedure consisting of the model or algorithm for the ALPHA. The data definitions and structure for the simulation are synthesized from the required data elements, their relationships, and their attributes in the ASSM.

3. Static Analysis

Since most requirements inconsistencies do not require simulation for their discovery, REVS provides several tools to statically check for completeness and consistency. They are able to detect deficiencies in the flow of processing and data manipulation stated in the requirements.

The first class of these tools is to check the structure of the R-NETS entered interactively, including one and only one start node, proper branching and rejoining of paths, and their proper termination.

The second class of tools checks the flow of data through the R-NETS. They check for definite and potential errors in data use.

The third class of tools checks for proper hierarchy in the specification. Definitions must be specified for all SUBNETs, that SUBNETs must not make reference to each other recursively, and that all ALPHAs and SUBNETs must appear on at least one R-NET.

4. Report Generation

In order to reduce the necessity of adding a new tool each time a specialized report or analysis is required, REVS provides the requirements engineer with a specialized tool known as the "extractor". The exactractor enables the user to control the scope of the analysis and content of the reports generated, not burdening him with format specification or the need to review tabular forms to extract information.

This system enables the user to subset elements in the ASSM based on some condition or conditions and then display the subset elements. The output produced is in RSL compatible, standardized format to which prepositions and punctuation are added to produce formal documentation.

The information the user desires to be retrieved is identified in terms of RSL concepts. For example:

```
SET A = DATA INPUT TO KALMAN-FILTER.

LIST A.
```

By combining and manipulating these sets it is possible to detect the presence and absence of data, trace references, and analyze interrelationships.

The extractor provides both reports for ad hoc inquiries and routinely generated special reports which enable managers to check for completeness and consistency, and perform automatic regressive testing.

V. PSL/PSA

A. INTRODUCTION

The survey discussed in the next chapter revealed that many commands in the U.S. Navy interested in RSL have used the Problem Statement Language/Problem Statement Analyzer (PSL/PSA) system. For that reason a brief overview of this system will presented, as well as a section that deals with some of the drawbacks of both SREM and PSL/PSA.

B. PSL/PSA OVERVIEW

PSL/PSA [15] was designed to provide an improved approach to system design. This approach is based on the premise that more effort and attention should be devoted to the front end of the process where a proposed system is being designed by the potential user; that since large amounts of information are being handled, a computer should be used; that computer-aided approaches to system development must start with documentation.

The system is based on a counterpart of RSL, namely PSL or Problem Statement Language. It is based on a model of a general system and also on the specialization of the model towards information systems .

Much like SREM, PSL defines a set of OBJECTS which have PROPERTIES and PROPERTY VALUES and their interconnections are referred to as RELATIONSHIPS. PSL also takes into account timing and volume considerations.

The intent of PSL is to separate the definition of user requirements and the processing solution of these requirements [16]. If the two were carried out in concurrently, requirement changes in the future may not be accomodated due to a firm design. Therefore, PSL does not presuppose solutions, it only states requirements.

The second half of the system, the Problem Statement Analyzer or PSA, performs basically the following functions:

1. Data Collection

   Several intermediate outputs of the PSA include checklists for deciding what additional information is required.

2. Analysis

   A variety of analyses previously performed manually can be handled by PSA, including static analysis of the entire developed system.

3. Design

   PSA allows data to be manipulated more extensively by the designer.

4. Evaluation

   PSA can perform computations on volume or work measures from data in the problem statements [15].

The PSA also serves as a report generator, including narrative description, lists, tables, arrays, matrices, diagrams, and charts. The PSA can produce reports on what changes have been made in the database, reference data items of similar type or property, or produce reports of analytical nature such as gaps in information flow, similarity of inputs and outputs, and the dynamic nature of the system [15].

C. COMPARISION OF SREM AND PSL/PSA

One of the difficulties in the area of RSL is that there has been no in-depth comparative studies of the effectiveness of the various systems and methodologies. The main reason is readily apparent: such an endeavor would be costly in terms of both time and money and the criteria for judging overall effectiveness and usefulness would be difficult to develop. However, there are some practical aspects of SREM and PSL/PSA that bear some scrutiny.

1. Transportability

SREM, at the moment, is highly machine dependent due to memory hierarchy mapping and that the bulk of the system operates with approximately 60,000 lines of PASCAL. SREM is presently operating on Texas Instruments Advanced Scientific Computer (ASC) and certain models of Control Data Corporation's CDC 7600. Work is presently underway to make SREM compatible with Digital Equipment Corporation's VAX-11 system.

PSL/PSA does not have this problem. It is written in standard FORTRAN, making it compatible with a wide range of computers.

2. Graphics

With SREM, the use of the CALCOMP plotter imposes a severe limitation on the number of elements that can be drawn. However, the on-line graphics package, along with the features discussed in the previous chapter, have demonstrated good editing capabilities and fast turn around [17].

PSL/PSA has some strict limitations in graphics. First is its representation of functional flow diagrams, called "process-chains" (Fig. 6 [17]). This type of representation cannot show all types of logic branching, such as IF-THEN-ELSE type constructs. One has to refer back to the formatted problem statements to determine the logic being used. Feedback cannot be represented as well [17].

PSL/PSA can, however, produce a "picture-report" (Fig. 7 [17]), which is a partitioning of the various processes. The picture-report can show what the process is part of, inputs and outputs, and the entities the process uses or derives. A description of these items can be found in the formatted problem statements. This report can be very useful to program designers [17].

It should also be mentioned that PSL/PSA utilizes only a line printer for graphics output and that development work is being done to interface both plotters

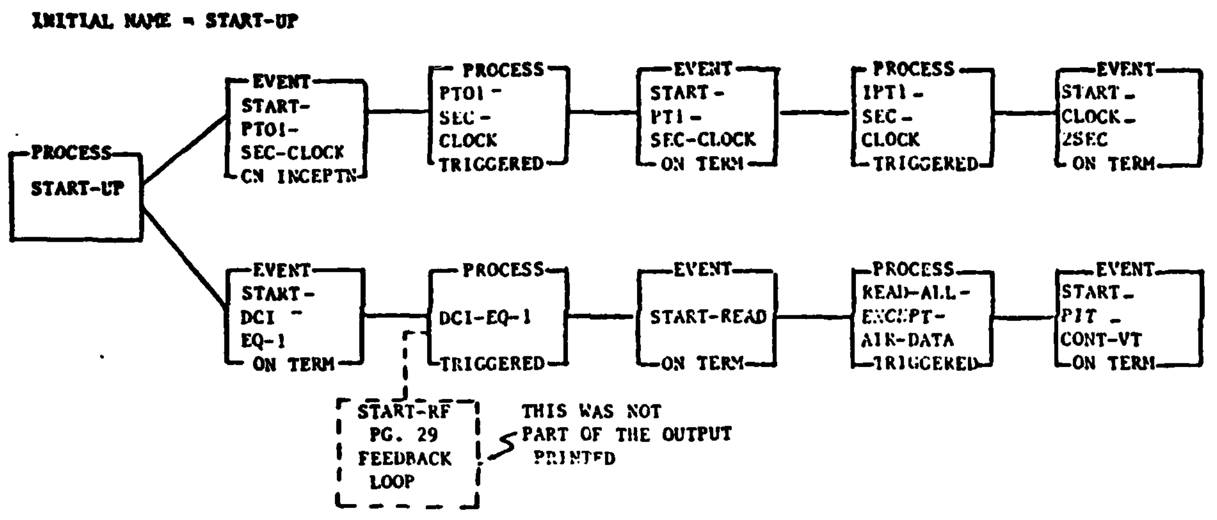

INITIAL NAME = START-UP
PROCESS START-UP
EVENT START-PT01-SEC-CLOCK CN INCEPTN
PROCESS PT01-SEC-CLOCK TRIGGERED
EVENT START-PT1-SEC-CLOCK ON TERM
PROCESS IPT1-SEC-CLOCK TRIGGERED
EVENT START-CLOCK-2SEC ON TERM
EVENT START-DCI-EQ-1 ON TERM
PROCESS DCI-EQ-1 TRIGGERED
EVENT START-READ ON TERM
PROCESS READ-ALL-EXCEPT-AIR-DATA TRIGGERED
EVENT START-PIT-CONT-VT ON TERM
START-RF PG. 29 FEEDBACK LOOP
THIS WAS NOT PART OF THE OUTPUT PRINTED

Figure 6.
Process Chains

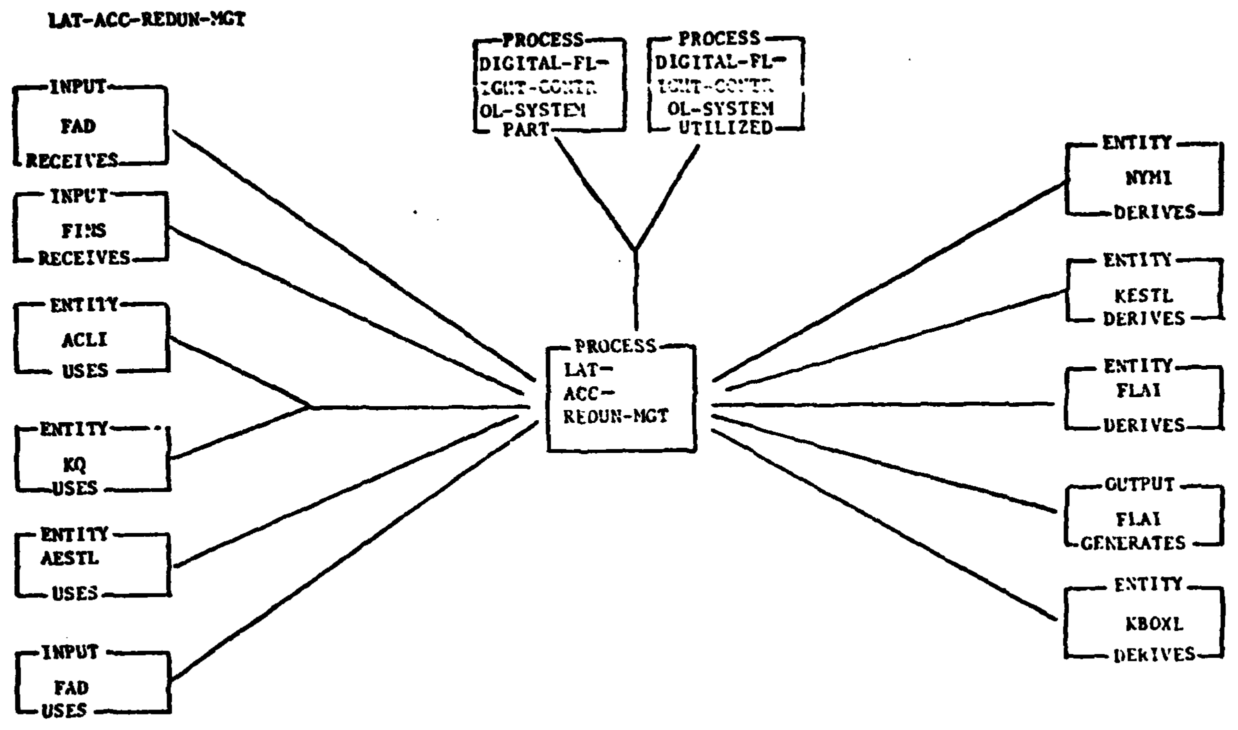


Figure 7.
Picture Report

graphics terminals. The use of line printers for the graphic output has caused difficulty in readability.

3. Simulation

SREM's static and dynamic capabilities were described in the preceeding chapter.

PSL/PSA, at present, has no simulation capability, but development work is underway to implement this feature utilizing SIMSCRIP II.5. [fur].

4. Other Considerations

As described previously, SREM was developed for large, real-time systems. The approach taken in the development of PSL/PSA was more universal: the system was aimed towards utilization by a wide range of users.

VI. UTILIZATION OF RSL IN THE U.S. NAVY

A. INTRODUCTION

A sizeable portion of the research behind this thesis was spent in conducting a telephone survey of various naval centers engaged in research and development. The centers contacted were:

1. Naval Research Laboratory, Washington, D.C.
2. Naval Air Development Center, Warminster, Pa.
3. Fleet Combat Direction Systems Support Center, San Diego, Ca.
4. Naval Oceans Systems Center, San Diego, Ca.
5. Naval Surface Weapons Center, Dahlgren, Va.
6. Naval Underwater Systems Center, New London, Conn.

The purpose of the survey was to ascertain the current level of utilization of requirements statement languages in system design and development. The personnel contacted were questioned as to which RSL and/or methodology was currently being employed, to what type of project it was being applied, perceived or proven successes and failures, how much interest has been expressed by higher authority, and their personal assessment as to the future of such tools in system specification, design, and development. A summary of the findings follows, however, the views expressed should not and cannot be taken as the official position of the individual commands.

B. NAVAL RESEARCH LABORATORY (NRL)

Other than some work performed for the Applied Physics Laboratory, The Johns Hopkins University, in 1978, utilizing SREM, there has been little interest expressed in RSL per se. However, Heninger et al. [18] have advanced the notion of developing a disciplined methodology in order to develop clear, concise requirements specifications through their work in redesigning and rebuilding the operational flight program for the A-7 aircraft.

It is their contention that it is necessary to approach such a problem by formulating questions before answering them, rather than being influenced by available information, separating concerns, and using precise notation [7]. From these basic principles they developed their disciplined approach which is more fully discussed in [7] and [18].

C. NAVAL AIR DEVELOPMENT CENTER (NADC)

NADC was introduced to RSL when it was directed by Naval Air Systems Command (NAVAIR) in 1978 to install and utilize SREM in conjunction with the CV/TSC project (since redesignated CV/ASWM), an effort to develop a computer-based tactical support center for S-3A aircraft to be integrated with the Naval Tactical Data System (NTDS).

Problems with the utilization of SREM were caused by its being introduced too late in the development phase. Personnel were not comfortable with it and there seemed to be a lack of unanimity among these same personnel as to whether or not the SREM approach to system design was viable.

Though no further projects have utilized SREM, several internal studies have been conducted at NADC aimed at determining its feasibility for future projects. The interim findings have suggested that SREM or some similar methodology should be more actively incorporated into the requirements definition phase of system development. The Air Force's Rome Air Development Center (RADC), Griffiss Air Force Base, New York, will soon send personnel to NADC for developmental work with SREM.

D. FLEET COMBAT DIRECTION SYSTEMS SUPPORT CENTER (FCDSSA)

FCDSSA has looked closely at the problem of requirement level documents as they are currently developed and at the use of methodologies and automated tools in defining and analyzing requirements for tactical data system software. Their study of requirement level documents revealed the lack of conformity in terminology, such as:

- different words and phrases used to convey the same meaning.

- same words and phrases used to convey different meanings.

- slightly different words and phrases used with only slightly different meanings.

- different disciplines such as navigation, sensors, aviation, fire control, etc. having different terminology, complicating their integration into the overall system.

Additionally, too often the applicability to subsystems; the conditions, external and internal, under which the requirements apply; and the duration of their applicability are ill-defined.

FCDSSA feels that any proposed methodolcgy should include:

- disciplined requirements statement language.

- extensive use of graphics to facilitate communication.

- model building techniques for verifying completeness.

Above all, it is felt that it is the methodology, not the tools employed, that is of the greatest importance.

Since early 1978, FCDSSA has evaluated several systems utilizing RSL, including SREM and PSL/PSA. It noted the strong and weak points of each system and decided that none provided the flexibilty, user interaction, and ease of use that it thought to be mandatory. Therefore, it embarked in late 1978 on the development of its own requirements language analyzer, named CORVAIR. The eventual aim is to produce a system with a highly extensible language that can be configured to suit the needs of the individual and that will ultimately produce source code from the requirements automatically.

It was mentioned that the requirements developed for a project utilizing a system using RSL were not accepted by a contractor; it was felt by the contractor that the system was already designed by FCDSSA. It was the opinion of the person contacted at FCDSSA that an effort is needed to educate all parties in the government and civilian sectors as to exactly what the purpose of RSL developed specifications serve.

E. NAVAL OCEAN SYSTEMS CENTER (NOSC)

The System Design Laboratory at NOSC found PSL/PSA to be of great use in enforcing discipline in the way requirements specifications are written. As an example, they checked the specifications of the NTDS Model 4 software for FCDSSA, using PSL/PSA. Their analysis uncovered over 200 occurances of ambiguous, undefined, or inconsistent statements.

A feature of PSL/PSA that was very well received was the ability to store the developed requirements specifications in a database and the ability to partition specifications in order to determine the effect that any modification would have on the overall system.

NOSC has not received much direction as to the use of PSL/PSA or any other RSL. NOSC has, however, been a strong proponent of such a system and has conducted seminars for government and civilians in the San Diego area. The personnel involved feel that it is an area that should be actively pursued and developed.

One of the problems noted was the difficulty of mapping RSL developed requirements into the structure required by SECNAVINST 3560.1.

F. NAVAL SURFACE WEAPONS CENTER

The Naval Suface Weapons Center is presently incorporating PSL/PSA into the life cycle support of the AEGIS combat systems. Their initial work has centered around the retrofitting of AEGIS specifications into PSL so as to verify and validate the system at least on a high level. It is hoped in the future the work will be focused on lower levels of the system to check the stimulus/response of individual modules and eventually investigate the automated generation of performance specifications.

PSL/PSA has been very well received by personnel at the center. They very much feel that this is the direction in which requirements definition in the system design should proceed. Briefings on this technique have been given to officials from Washington, D.C. and they, in turn, have expressed some interest in its development.

Problems noted by the center were the need of educating personnel as to the techniques involved and the fact that SECNAVINST 3560.1 does not facilitate the use of RSL generated specifications because this instruction predates the development of RSL.

G. NAVAL UNDERWATER SYSTEMS CENTER (NUSC)

The experience at NUSC with RSL has been limited to the IBM Federal Systems Division's work on the Submarine Active Detection Sonar (SADS) project using PSL/PSA.

The project, as far as utilization of PSL/PSA, proved to be unsuccessful and was finally abandoned. The person contacted at NUSC listed as some of the problems:

- personnel at NUSC were not sufficiently familiar with PSL/PSA to fully appreciate its capabilites and peculiarities.

- due to security considerations and the fact that the host IBM 370 computer had to be shared with others, forcing third shift operations, there existed a time constraint on development work.

- the output produced was hard to understand.

- there were constraints imposed by SECNAVINST 3560.1 that could not be waived.

IBM Federal Systems Division was contacted for its views on the problems with the use of PSL/PSA and SADS. They noted:

- personnel were not sufficiently familiar with PSL/PSA.

- there was poor support for the tool as it had been only recently installed.

- adequate training was not received by personnel involved in the use of the system.

- there is extreme difficulty in attempting to translate PSL/PSA generated requirements into the narrative form required by SECNAVINST 3560.1.

The personnel contacted at IBM Federal Systems Division said that they felt that PSL/PSA would be of significant value on future government projects. They are confident that most of the problems experienced on the SADS project will be corrected.

H. SUMMARY

The survey conducted revealed several views that were expressed by the majority of the personnel interviewed. They were:

- the use of RSL has forced discipline in specification writing. As a consistency checker, it has uncovered numerous errors in critical documents.

- the concept and use of RSL should be continued and expanded in future projects.

- there exists an education gap in both the government and civilian industry as to the use of RSL. This is a problem that must be resolved so as to avoid the misunderstanding and misapprehension experienced in the past.

- strong management support is required to overcome the tendency by some to resist change, regardless of how proven a new technology may be.

- though not addressed in the above sections, the majority felt it would benefit the government to utilize RSL early in the conceptual phase of a project instead of introducing its use after the specifications have been written. A conversation with Dr. Teichroew, one of the prime developers of PSL/PSA, revealed that the vast majority of private sector users of his system use it from project inception.

- it is extremely difficult to translate RSL generated specifications into the form required by SECNAVINST 3560.1.

VII. RSL AND SYSTEM ACQUISITION

It is important to examine how RSL methodologies fit into the various rules, regulations, directives, and standards that currently govern systems acquisition in the Department of Defense and the U.S. Navy. It would be neither possible nor meaningful to examine every document dealing with this area, nor would it be possible to do an in-depth analysis of each. Rather, it is the intent of this chapter to look at some of the major points stressed in the above rules, regulations, etc. and determine whether or not RSL methodologies satisfy the letter and intent of these documents from both the government and the contractor points of view and to consider changes which may be necessary to better incorporate the capabilites of RSL methodologies.

A. OMB CIRCULAR NO. A-109

On April 5, 1976, the Office of Management and Budget issued Circular No. A-109, "Major Systems Acquisition"[19], to the heads of all Executive departments in the government. The purpose of A-109 was to give strict guidance in the acquisition of major systems. It stressed: (1) justification of the acquisition based on mission need, not the perceived need of new hardware, software, etc., (2) competitive development of alternate solutions to solve the mission need, (3) tradeoffs between cost, performance, and production schedules, (4) ensuring adequate test and evaluation of the new system, and (5) development of a sound acquisition strategy, looking at the entire life cycle.

B. DEPARTMENT OF DEFENSE DIRECTIVES 5000.1 AND 5000.2

The Department of Defense's implementation of A-109 was Department of Defense Directive (DODD) 5000.1, "Major Systems Acquisition", and DODD 5000.2, "Major Systems Acquisition Process". In both of these directives are areas in which an RSL methodology may prove beneficial.

1. Technology Base

DODD 5000.1 tasks each DOD Component Head, such as the Secretaries of the Army, Navy, and Air Force, with advancing technology in both product and manufacturing technology to support future system development. It is recommended that the methodology employed in designing and developing software should certainly be incorporated into this base. As DOD's level of experience with the use of a formal methodology in the design and development of software grows, it certainly is quite feasible that modifications to the methodology may be warranted, certainly in the critical area of real-time systems. This inclusion in the technology base ensures a greater probability of wide dissemination of the methodology to those agencies and contractors involved in the system acquisition process.

2. <u>Competitive Exploration of Alternative Solutions</u>

DODD 5000.1 and DODD 5000.2 state that after a mission need has been established and approved there will be a competitive exploration of alternate solutions to the need. Participation in this exploration is open to industry, educational institutions, and government facilities.

Though industry and educational institutions are considered to be the primary sources of solutions, this in no way should lessen the contribution that government laboratories and facilities can make through use of an RSL methodology and a corporate history of lessons learned.

A hypothetical case might be the total replacement of the Naval Tactical Data System (NTDS) during the 1990's due to system obsolescence. By this time there will have been a large database built concerning the performance problems, acquired from user reports in the past and from the evaluative study required to establish the mission need of a replacement system.

A methodology such as SREM might prove beneficial to an on-going, evaluative study as it enables personnel to determine the effect of additional or modified requirements on a system such as NTDS. The lack of an expeditious and efficient handling of any new threat or threat scenario by the system could be determined as far as the present hardware and software configuration is concerned. These findings, coupled with the known problems in the development of the old system, serve as a solid foundation for the

Request for Proposals (RFP) that is sent to interested contractors, soliciting alternate solutions.

The RFP cannot presuppose system design but it should accurately reflect the user's requirements of the system. The contractors' proposals can be no better than the RFP on which they are based.

The government laboratory that undertakes the development of an alternate solution to the mission need should first of all be totally divorced from the group that developed the RFP so as to preclude the possibility of a prejudicial view of the system, which may stifle the creativity of the system designers, and also to ensure fair competition among the various parties involved.

The government facility may have an advantage in that it should have a better opportunity to evaluate the operational environment in which the system will be deployed. Since in the case of NTDS the government facility would in all likelihood be a Navy command, the personnel involved should have among them those who fully understand the functions of the Combat Information Center (CIC) in a wartime environment. This alone should improve the human engineering aspect of the system design, a facet too often overlooked or misunderstood, especially in the stressful situation of actual combat. Not only is a system that works critical, but also a system that can be effectively interfaced by personnel of various ranks, educational and experience levels. An RSL methodology can enable the designers to take this into consideration as the sytem design develops since all inputs, outputs, and human interfaces become highly visible.

C. DEPARTMENT OF DEFENSE DIRECTIVE 5000.29

DODD 5000.29, "Management of Computer Resources in Major Defense Systems" [22], addresses the problem of management and control of computer resources during the development, acquisition, deployment, and support of major defense systems.

This directive has a significant impact on software. It mandates that the software design (specifications) be validated (demonstrated that it satisfies all current stated requirements of the system) during the Concept Formulation and Program Validation phases of system development, prior to the Defense Systems Acquisition Review Council (DSARC) II. (DSARC II rules on whether or not to permit full-scale engineering development of a proposed system).

Other points emphasized are that correctness of software, reliability, integrity, maintainability, ease of modification, and transferrability are major considerations in the initial design.

The above paragraph contains what must still be considered moot points: these requirements have yet to be defined in a manner by which a universally accepted criteria for evaluation of these requirements can be established. However, the validation requirement should serve to force the issue of requirement and specification visibility. A methodology such as SREM can enable the contractor to accurately validate his specifications against the stated requirements and verify that his system, as far as the specifications are concerned, will function properly.

The cognizant naval agency could also use SREM to validate the contractor's specification. This, however, may prove troublesome if the specifications are not written in RSL format as there may not be an accurate translation of the specifications from narrative form to RSL form.

The criticality of the validation process cannot be overemphasized. It is the last point in the acquisition process in which major changes can easily be implemented into the system design. Once full-scale engineering development is initiated, the Navy effectively reduces its design control. Therefore, the use of an RSL methodology can help ensure proper validation of system design.

D. DEPARTMENT OF DEFENSE INSTRUCTION 5010.21

DODD 5010.21 [23] is entitled "Configuration Management Implementation Guidance". Configuration management is a discipline applying technical and administrative direction and surveillance to (1) identify and document the functional and physical characteristics of a configuration item (hardware/software that satisfies an end use function), (2) control changes to those characteristics, and (3) record and report change processing and implementation status.

As previously discussed, both SREM and PSL/PSA can perform certain configuration management tasks, including "locking-in" selected portions of the design to prevent further change, and they will also generate reports as to changes made to the database.

E. MILITARY STANDARD 1679 (NAVY)

Military Standard (MIL-STD) 1679 (NAVY), "Weapon System Software Development", [24] was developed to reflect the need to have more stringent control in the development of software for weapons systems. The main reasons for this were the criticality of performance inherent in such systems, a changing operational environment necessitating changes to the system, and the high life cycle cost.

Appendix A contains Chapter 5 of this MIL-STD entitled, "Detailed Requirements". Because of their capabilites discussed in previous chapters, it is felt that systems such as SREM and PSL/PSA directly aid the contractor in meeting the requirements imposed by the following sections of Chapter 5:

5.1.2.5.b	Block Diagrams
5.1.2.5.d	Function Description
5.1.2.6	Detailed Functional Requirements
5.2.2.3	Program Functional Flow
5.4.2	Naming
5.4.4	Narrative Description

5.5	Program Production
5.5.6.2	Cross-reference Listing
5.9.1.4	Program Design
5.11	Configuration Management
5.11.1.2	Documentation Identification

There are, however, two sections which should be considered for modification so as to better utilize a system such as SREM.

The first is section 5.2, "Program design requirements". Chapter IV included a discussion of the problems inherent with the traditional functional hierarchy approach to the development of requirements. SREM does not design in such a manner, it utilizes a flow orientation to the problem. This is presently not compatible with the above section.

The second is section 5.4.6, "Flow charts". SREM has the capability of producing detailed functional-flow diagrams. These diagrams can give a clear, concise view of the system's operation and the interrelationship of the various functions; a very valuable visual aid. If a system is indeed developed utilizing SREM, functional-flow diagrams should be considered a deliverable item.

F. SPECIFICATION AND DOCUMENTATION STANDARDS

Perhaps the greatest conflict between RSL systems and the requirements imposed in the systems acquisition process is in the area of specification and documentation. One of the problems discussed in Chapter VI was that RSL generated specifications do not easily map into the structure and format required by OPNAVINST 3560.1, "Department of the Navy Tactical Digital Systems Documentation Standards" [25]. The same holds true for Military Standard 490, "Specification Practices" [26].

Appendix B contains an excerpt from MIL-STD 490 which deals with specifications applicable to development of computer programs.

Figure 8 lists the required inputs for a hypothetical engine monitoring system. Figure 9 lists the required processing flows for the same system. Both of these were produced by SREM and should be compared to sections 60.3.2.1 and 60.3.2.2 in Appendix B respectively. It is evident that RSL generated specifications are of a highly structured nature, whereas MIL-STD 490 is narrative dependent.

The chief complaint expressed by RSL users towards standards such as MIL-STD 490 is that such a standard imposes such a strict format that the structure of the system is lost, especially since a system such as SREM structures it designs uniquely.

For now, the above complaint should be considered one of a highly subjective nature. Some users have at their disposal a translator which transforms specifications of the form given in Figures 8 and 9 into the form required by MIL-STD 490, including narrative, with some degree of success. It will take both further use and refinements of

```
SUBSYSTEM:  ENGINE-MULTIPLEXER
  CONNECTED TO
    INPUT-INTERFACE:  MUX-INPUT
      PASSES
        MESSAGE:  ENGINE-MEASUREMENTS
          MADE BY
            DATA:  MEASUREMENTS
              INCLUDES
                DATA:  SENSOR-DATA
                  INCLUDES
                    DATA:  MEASURED-P1
                    DATA:  MEASURED-P2
                    DATA:  MEASURED-P3
                    DATA:  MEASURED-P4
                    DATA:  MEASURED-T1
                    DATA:  MEASURED-T2
                DATA:  SWITCH-DATA
                  INCLUDES
                    DATA:  MEASURED-S1
                    DATA:  MEASURED-S2
SUBSYSTEM:  ENGINEERING-STATION
  CONNECTED TO
    INPUT-INTERFACE:  FROM-ENGINEER
      PASSES
        MESSAGE:  ENGINE-SET-UP
          MADE BY
            FILE:  SET-UP-LIST
              CONTAINS
                DATA:  SET-UP-DATA
                  INCLUDES
                    DATA:  NEW-PARAMETERS
                    DATA:  NEW-VALUE
           MADE BY
             DATA:  COMMAND-TYPE
             DATA:  ENG-NO
         MESSAGE:  HISTORY-REQUEST
           MADE BY
             DATA:  COMMAND-TYPE
```

Figure 8.
Required Inputs

```
R-NET:  PROCESS-ENGINE-DATA
   STRUCTURE:
     INPUT-INTERFACE:  MUX-INPUT
     VALIDATION-POINT:  VP-1
     ALPHA:  VALIDATE-MESSAGE
     CONSIDER DATA:  DATA-VALIDITY
     IF (VALID)
        SELECT ENTITY-CLASS:  ENGINE SUCH THAT
        (CHANNEL-NUM = MONITOR-CHANNEL-NO)
        DO
            ALPHA:  UPDATE-HISTORY-FILE
            TERMINATE
        AND
            ALPHA:  COMPARE-TO-LIMITS
            CONSIDER DATA:  MEASUREMENT-STATUS
            IF (ALARM-STATE)
               ALPHA:  TRANSMIT-ALARM
               VALIDATION-POINT:  VP-3
               OUTPUT-INTERFACE:  TO-ENGINEER
            OR (WARNING-STATE)
               ALPHA:  TRANSMIT-WARNING
               VALIDATION-POINT:  VP-4
               OUTPUT-INTERFACE:  TO-ENGINEER
            OR (NORMAL)
               TERMINATE
            END
```

Figure 9.
Required Processing

the translators to instill confidence in the translaed requirements.

G. SUMMARY

The above sections are by no means a comprehensive review of the documents discussed, nor do they review all documents governing system acquisition. However, the above discussion points out the fact that RSL systems, for the most part, can be made to support the current system acquisition process, if the incompatibilities of RSL with existing military specifications and standards can be resolved.

VIII. RECOMMENDATIONS

A. INTRODUCTION

The Navy is in need of increasing its activities in the requirements definition area. DOD has provided development funds to the Air Force for URL/URA (a derivative of PSL/PSA) and to the Army for SREM [27]. The Navy currently has no development projects of this nature.

The research conducted by the other services may be of benefit to the Navy, but there is no guarantee of universality in its application. Due to differences in weapon systems requirements and overall management philosophies, it is highly unlikely that inter-service transfer of technology could occur without undue modification. One Navy user of URL/URA found it unsatisfactory for Navy applications. As subjective as this opinion may be, it points out the fact that, much like aircraft, it is nearly impossible to satisfy two services with a single system.

The Navy needs to take corrective action to systematically improve its procedures for the development of software. The first step recommended is to hold a major conference with all facilities within the Navy involved in software/system design, including project offices, represented. Some of the area that should be dealt with are discussed in the following sections.

B. PROBLEM IDENTIFICATION

It is quite important to initially identify all problems that currently exist in the development of requirement specifications and the application of automated tools/methodologies. Those problems cited in Chapter VI are only a small fraction of the those existent today. Through their proper identification, a strategy can be evolved to develop solutions.

C. AUTOMATED TOOL AND METHODOLOGY EVALUATION

At present, there has been no comprehensive, comparative study of the major automated tools and methodologies that currently exist in this field. Initiation of such a study should seriously be considered by the Navy.

The study should be initiated under the premise that no one tool or methodology will entirely satisfy the needs of all projects. Real-time combat systems and ADP systems are almost totally divergent in their system requirements. Though current experience shows that, at a minimum, a disciplined methodology of some form is required throughout the entire spectrum of software-related projects, it is a question of applying a particular methodology or automated tool where it will give the greatest return in terms of improved definition of requirements specifications. Acceptance of any new technology comes mainly through demonstration of superior results.

The tools and methodologies chosen for application in a particular area should meet known requirements, have a capacity for evolutionary growth, and have a reasonably long expected lifetime. Above all, it must be understandable and suitable for training [28]. The fact that tools and methodologies developed by highly trained, highly educated personnel do not guarantee successful application by personnel of varying backgrounds should not be overlooked. The members of the evaluation group must reflect this diversity.

D. ACCEPTANCE AND TRAINING

Some of the reasons stated by Wolverton [28] for personnel not using tools in general are that they see no benefit to them, lack of understanding of the tool, perceived high risk of failure, management coercion, and lack of time to experiment with the tool due to schedule pressures.

The first step in gaining acceptance of a tool or methodology is in total management support. Though management obviously cannot issue an edict mandating its immediate use with the expectation of immediate results, it can, nonetheless, provide firm guidance in its assimilation into the overall design process.

The training of the ultimate end-users should not only provide a thorough understanding of the tool or methodology, but should also be directed towards instilling confidence in the user as to his or her ability to use the tool or methodology.

Time is rarely in the favor of any project; therefore, the initial use of such a tool or methodology should be on a project which does not have great pressures of time and money. Through a systematic introduction, the tool or methodology will be afforded a better chance to succeed or fail on its own merits, not the perceptions of the users.

E. OTHER CONSIDERATIONS

The adoption of a tool or methodology for application in a particular area will certainly raise a myriad of questions that cannot be dealt with in this thesis, but as an example, take the hypothetical case that SREM is adopted as the standard automated tool for weapon sytems software development. Should this standard be imposed on contractors who wish to bid on future contracts in this area? If not, should the Navy train personnel in the various techniques used in industry so as to facilitate the liason between project office and contractor? Can the documentation and specification standards be modified so as to allow RSL generated specifications to be submitted in their structured form? These and other questions may have to be dealt with as the state of the art in software technology advances.

IX. CONCLUSIONS

This thesis has discussed some of the problems inherent in requirements specifications as they currently are developed and has looked at an evolving, disciplined approach to the problem in the form of requirement statement languages and systems. The promising, if not proven, automated tools utilizing requirement statement languages, SREM and PSL/PSA, have been shown to have capabilities that may prove to be of great value in systems acquisition. They have also been shown to have some drawbacks as well.

Also discussed in this thesis are some suggestions as to how the Navy should approach this technology, such as evaluation of these and other tools and methodologies, their incorporation into projects where the benefit would be greatest, and gaining the acceptance of those who would actually be required to use such systems.

Above all, this thesis has stressed that these types of tools and methodologies need to be seriously considered by the Navy as a possible solution to some of its problems in systems acquisition.

APPENDIX A - MILITARY STANDARD 1679

5.1 Program performance requirements. The contractor shall determine the detailed program performance requirements for the weapon system software. The contractor shall utilize the basic descriptive requirements and design information provided by the procuring agency to create the program performance requirements. This information may be augmented by studies, analysis, visits to operational units, and surveys as necessary. The program performance requirements are subject to the approval of the procuring agent.

5.1.1 Supporting information. The contractor shall utilize, as a minimum, that of the following supporting information which is available to determine the program performance requirements:

a. System-level performance requirements.
b. System-level design specifications.
c. Equipment design specifications.
d. Interface design specifications.
e. Operational standards, doctrine and tactics.
f. System design standards.

5.1.2 Analysis. In determining the performance requirements, the contractor shall investigate and analyze in detail all areas relating to the performance requirements of the weapon system software.

5.1.2.1 Mission areas. The contractor shall investigate the mission areas, primary and secondary, and supporting tasks of the operational user or platform for the weapon system.

5.1.2.2 Functions. The contractor shall define the major functions or groupings of the program necessary to meet the system performance requirements.

5.1.2.3 Applicable documentation. The contractor shall identify all documents which define or constrain the program performance requirements. Definitions of applicable terms and abbreviations not consistent with or not included in reference document 2.1.c shall be indicated and defined by the contractor.

5.1.2.4 Weapon system description. The contractor shall examine the relationship of all components in the weapon system which affect the program performance requirements or the computer program. He shall determine how the computer program interfaces with other components to perform required functions.

a. Peripheral equipment identification. The contractor shall identify all equipment with which the program will interface.

b. Interface identification. The contractor shall identify all other digital programs or systems with which the program will interface.

5.1.2.5 Functional description. The contractor shall analyze the major functions and the functional relationships of the program with interfacing equipments and other programs.

a. Equipment descriptions. The contractor shall identify the requirements imposed on the program by each interfacing equipment, the purpose of the equipment, and the use of options and controls.

b. Block diagrams. The contractor shall generate diagrams of equipment/program relationships with internal and external data flow.

c. Intersystem interface. The contractor shall determine the interfaces with other systems and shall be cognizant of the performance requirements and design specifications of all systems which will interface with the system under development. Each contractor shall be aware of the purpose of the interface and the data to be exchanged. Data quantity, frequency, rate, format, content, scaling requirements and conventions shall be developed. In fulfilling this assignment, the contractor may be tasked to participate with other development contractors as a team to design the inter-system interfaces so that the performance requirements of all systems are met. If interface conflicts are uncovered such that an individual system's ability to perform in accordance with its requirements is adversely affected, the interface design team shall recommend to the procuring agency the necessary modifications to the systems or their interface to overcome the deficiency. If no solution can be agreed upon, the team shall recommend modification of the system performance requirements to the procuring agent.

d. Function description. The contractor shall establish the performance of each function supported by the program, its purpose, and functional design.

5.1.2.6 Detailed functional requirement. The contractor shall delineate the performance of each function by detailing its narrative, logical, and mathematical descriptions.

a. Inputs. The contractor shall define all inputs (external and internal) including their sources, method of insertion, quantity, timing, range and scaling.

b. Processing. The contractor shall generate textual and, as appropriate, mathematical descriptions of the processing requirements of each function, including functional parameters and geometric diagrams.

c. Outputs. The contractor shall define all outputs (internal and external) including their method and timing, meaning, format, destinations, range and scaling.

d. Special requirements. The contractor shall identify all requirements imposed by higher-level constraints or by exigencies of the function.

5.1.2.7 Adaptive parameters. The contractor shall identify those parameters which reflect the system environment, system parameters, and system capacities, and which can be modified without altering the logic of the operational function.

5.1.3 System resources. The contractor shall define the computer memory, computer processing time and input and output resource budgets and the projected utilization for the weapon system. If the weapon system under development has more than one digital processor, the contractor shall define these resource values for each digital processor.

5.2 Program design requirements. The computer program design shall be developed from the program performance specification, and shall comply with other design constraints and standards as specified by the procuring agency. The software development shall be a top-down process. The design shall be a hierarchical structure of identifiable programs, subprograms, modules, procedures and routines. The highest level of control logic resides at the top of the hierarchy; the computational or algorithmic functions reside at the lower levels. The contractor shall define the assumptions, the programming approach for implementing the computer program and shall define the program architecture. The program design shall be subject to review by the procuring agency.

5.2.1 Supporting information. The contractor shall utilize, as a minimum, that of the following supporting information which is available to determine the program design:

a. System operational design documents.

b. Program performance specification.

c. Interface design specifications.

d. Programming reference manuals.

e. Equipment technical manuals.

f. Specified programming standards and conventions.

g. Specified utility/support software.

5.2.2 Computer program design analysis. In determining the detailed computer program design, the contractor shall investigate and analyze in detail the following areas relating to the computer program.

5.2.2.1 Applicable documentation. All documents which constrain, define, or influence the program design shall be analyzed. The contractor shall define all design terms and abbreviations used to describe the program design.

5.2.2.2 Functional allocation. The allocation of functions and tasks to be performed by the subprograms and a functional description of items, inputs, outputs, and processing to be performed shall be considered and subsequently defined. All performance requirements shall be satisfied in their entirety in this allocation.

5.2.2.3 Resource allocation and reserves. Memory storage and processing time for each subprogram shall be determined. Total system memory and processing time reserves of at least 20 percent shall exist at the time of program acceptance by the procuring agency.

5.2.2.4 Program functional flow. The flow of program data and control in all required modes of program operation shall be determined.

a. Program interrupt control. The source, purpose, type, predicted rate of occurrence, and required control response for each external and internal interrupt shall be determined from the analysis.

b. Subprogram reference control. The control logic, assignment of priorities, and permissible cycle times for each subprogram shall be determined from the analysis.

c. Special control features. Unique control requirements which affect the design of the control logic shall be identified.

5.2.2.5 Design constraints. The constraints of the specific programming language to be used; the constraints of the specific compiler, monitor, loader, librarian to be used; the capabilities of specific debug and utility aids for the program production; and the mnemonic labeling conventions required shall be defined by the contractor.

5.2.2.6 Data base design. All data used by two or more subprograms shall be taken into account during the computer program design.

5.2.3 Intersystem interface. The contractor shall determine the interfaces with other systems and shall be cognizant of the performance requirements and design specifications of all systems which will interface with the system under development. Each contractor shall be aware of the purpose of the interface and the data to be exchanged. Data quantity, frequency, rate, format, content, scaling requirements and conventions shall be developed. In fulfilling this assignment, the contractor may be tasked to participate with other development contractors as a team to design the inter-system interfaces so that the performance requirements of all systems are met. If interface conflicts are uncovered such that an individual system's ability to perform in accordance with its requirements is adversely affected, the interface design team shall recommend to the procuring agency the necessary modifications to the systems or their interface to overcome the deficiency. If no solution can be agreed upon, the team shall recommend modification of the system performance requirements to the procuring agent.

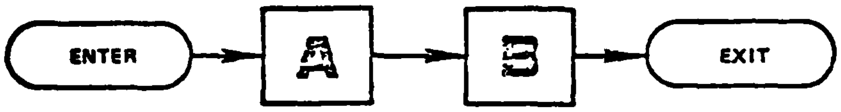
5.3 Programming standards. The following coding and logic standards shall apply to the implementation of subprograms.

5.3.1 Control structures. Programs shall be designed using only the five basic control structures presented in figure 1. They are: The SEQUENCE of operations (assignment, add,...), IF THEN ELSE (conditional branch to one of two operations and return), WHILE DO (operation repeated while a condition is true), DO UNTIL (operation repeated until a condition is true) and CASE (operation which provides the transfer of program control to a specific location within a compile-time system).

5.3.2 Entry-exit structure. Each module, subprogram, routine, or procedure shall have a single entry and single exit structure. (See figure 2.)

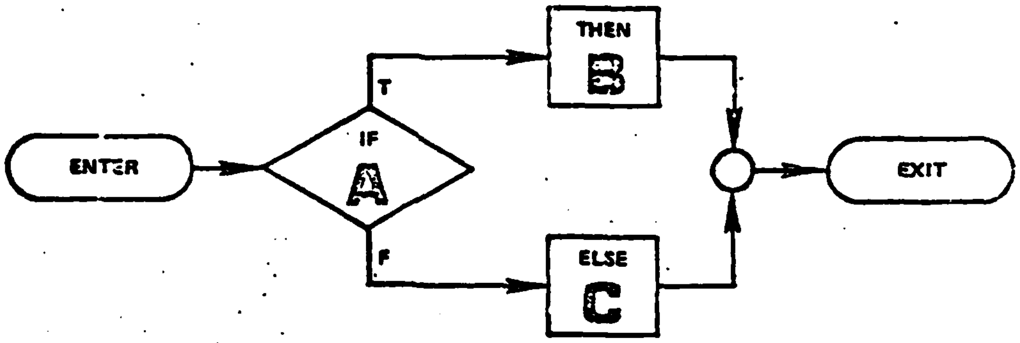
5.3.3 Source code segment includes/copy. When repetitive segments of source code are required in the program being developed, they shall be coded only once as a structural source code block, thereafter being referenced/utilized upon each occurrence by appropriate INCLUDES or COPY features, or constructs of the source HOL compiler. These included/copied segments shall be written in HOL only. Any program logic within a given structural segment shall utilize only those control structures specified in paragraph 5.3.1. For maximum memory efficiency, common routines or procedures should be used instead of included/copied source code blocks whenever practicable.

Figure 1A.
SEQUENCE.



Control flows from process A to the next in sequence, process B.

Figure 1B.
IF THEN ELSE.



The flow of control will return to a common point after executing either process B or C. A predicates the conditional execution. If control is to skip a process pending the condition of A, then the flow chart can be modified thusly: (See next page)

FIGURE 1. Control Structures.

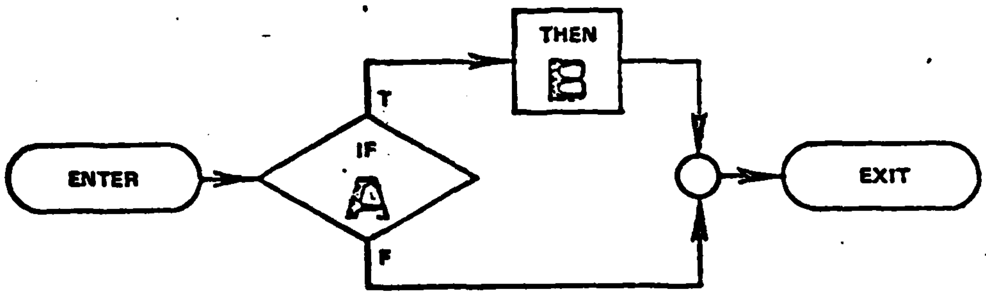
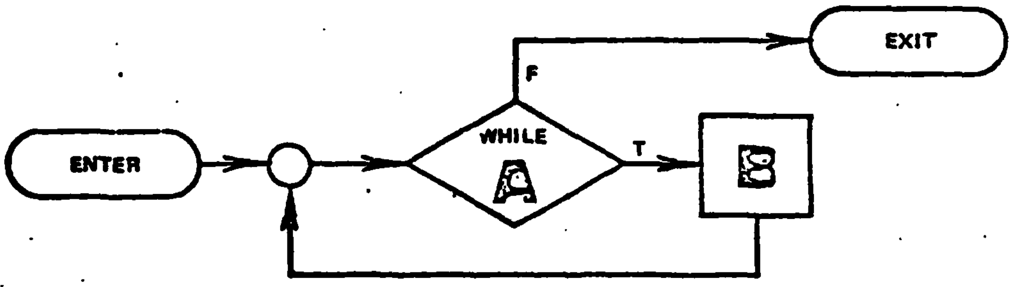


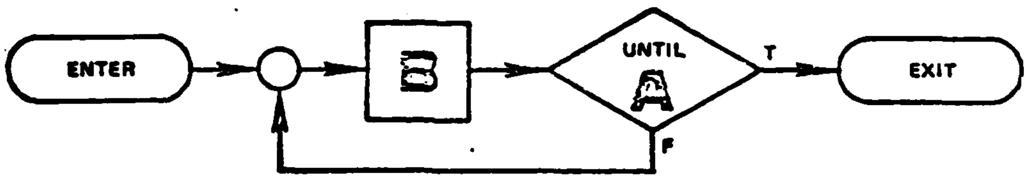
FIGURE 1B. (Continued)

FIGURE 1C. WHILE DO.



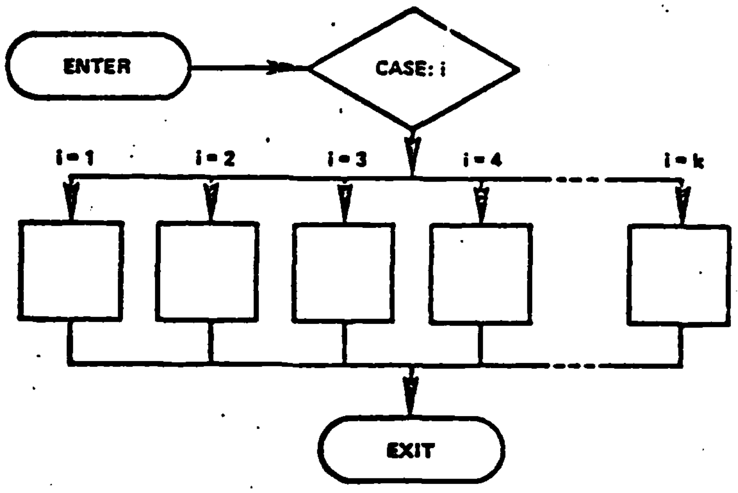
The WHILE DO structure is a loop, in which the condition A is evaluated. If found to be true, then control is passed to process B, and then condition A is evaluated again. If condition A is false then control is passed out of the loop.

FIGURE 1D. DO UNTIL.



The DO UNTIL structure is similar to the WHILE DO — except that the test of condition A is performed after process B has executed. Thus the DO UNTIL loop will be performed once regardless of the value of condition A.

FIGURE 1E.
CASE.



Control is passed to process 'K' based on the value of i. Structured programs of any degree of complexity can be built up, if they can be broken down into individual components.

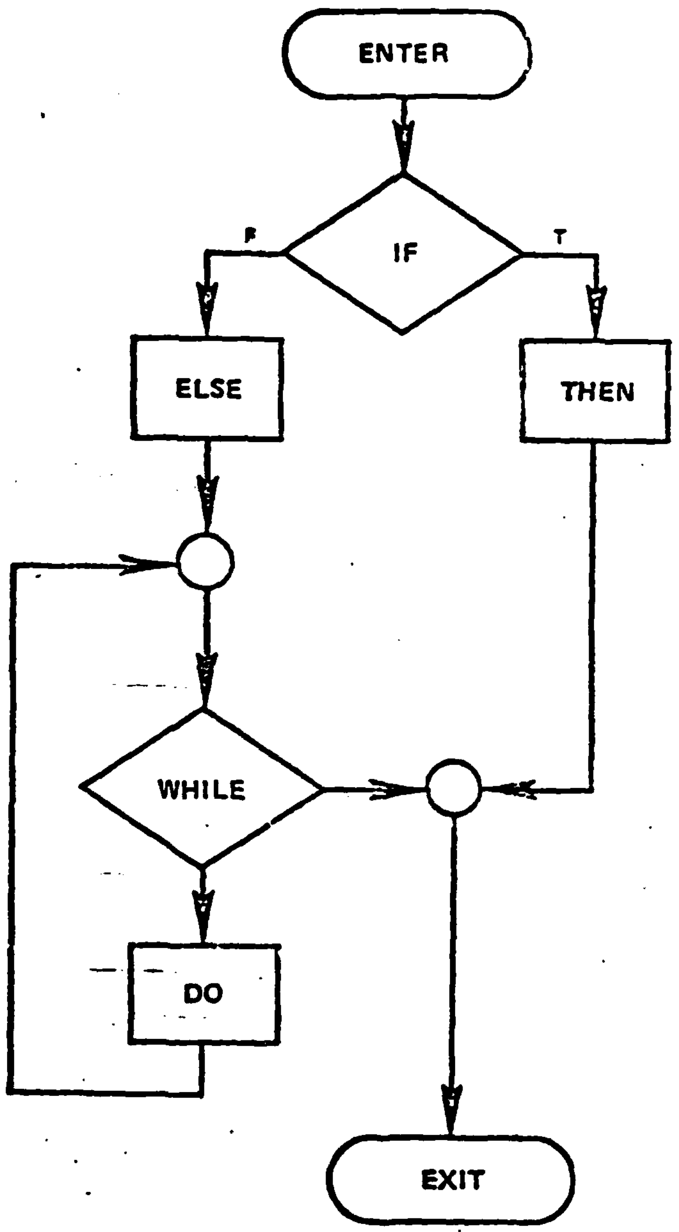


FIGURE 2. Nesting of Control Structures.

5.3.4 Program traceability. Programs shall be designed and constructed such that upon interrupt or termination, the values of the various parameters, indices, and other local variables as of the last usage are recoverable.

5.3.5 Self-modification. Program self-modification of instructions during execution shall be prohibited.

5.3.6 Recursive programs. Recursive procedures or programs shall not be used unless the target computer has a stack oriented architecture.

5.3.7 Size. The procedures or routines which make up a module or subprogram shall not exceed an average of fifty executable HOL statements per procedure or routine. Each independently executable HOL statement, whether free-standing or included within a complex statement, counts as one of the fifty.

5.3.8 Branching. Branching statements (GO TO's) are to be avoided if possible, and used only with the approval of the procuring agency. Branching statements, if approved, shall only pass control to a statement that is in the same procedure or routine. Each GO TO must pass control only forward of its point of occurrence. Backward jumps generated by the compiler are permitted. Transfers from a procedure or routine shall only be to the entry point of another procedure or routine.

5.3.9 Relocatability. The software shall be built in the form of relocatable object modules.

5.4 Programming conventions. The following programming conventions shall be utilized in all weapon system software.

5.4.1 Symbolic parameterization. All values used in the weapon system software which are constant throughout the weapon system design but which may be affected by environment changes (e.g., sensor output limits, maximum range of weapons, maximum number of targets handled, data storage limits) shall be treated as symbolic parameters in the design. Duplication of symbolic parameters shall be minimized through use of common source of values. When duplication is necessary, common symbolic parameter identification nomenclature shall be used and comments will point to location of duplicates. Symbolic parameters shall be grouped at the beginning of each program. Comments shall provide a definition and the location of all parameters. Special symbolic parametric definition features of the high level language and compiler shall be used.

5.4.2 Naming. Naming conventions shall be uniform throughout the weapon system software.

5.4.2.1 Modules. Module names shall be uniquely chosen to identify the applicable function performed and the hierarchical logic structure in relation to other modules in the system being developed.

5.4.2.2 Data. Data names shall indicate the function of the data item.

5.4.3 Numerical conventions. Numerical conventions shall be established by the contractor so that they are uniform throughout the program.

5.4.3.1 Symbolic constants and variables. Constants and variables entering into numerical computations shall follow the constraints set forth in paragraph 5.4.1.

5.4.3.2 Mixed mode expression. Mixed-mode numerical operations shall be avoided whenever possible, but when determined to be necessary, they shall be completely described in comments.

5.4.3.3 Grouping. Parentheses or other subexpression delimiters shall be used where necessary to clarify the order of evaluation of compound expressions.

5.4.3.4 Significant digits. The number of significant digits as output shall not be greater than the number of significant digits as input. The effect of truncation performed shall be considered in applying this convention. Sufficient significant digits shall be used in calculations to yield a minimum of computational error, and rounding by the programmer shall not occur until the final computational step. The degree of computational error shall be analyzed to determine if systems accuracy requirements are fulfilled.

5.4.4 Narrative description. A narrative description shall describe the history and identify the functions of procedures and routines.

5.4.4.1 Abstracts. Each procedure and routine shall include at the beginning of the executable coding a textual description of its inputs, outputs, function or task, and algorithms; list other procedures or routines called; and list all calling procedures or routines. In addition to general explanations, to assist understanding, precise references to the appropriate statement labels and data-names shall be included in each descriptive abstract. Local, previously undefined data-names shall be described. The descriptive abstract shall define the allowed and tolerable range of values for all inputs and shall define the allowed and expected range of values for all outputs.

5.4.4.2 Identification. Each procedure and routine shall carry an identifying label-name indicating function and hierarchical structure. A history of the original and updating programmer names, the activity or commercial company name and the activity or company division code or billet identifier with dates completed shall be included.

5.4.4.3 Statement comments. In order to facilitate program comprehension, comment statements shall be used throughout the program code. Comment statements are non-executable (i.e., those which have no effect on computer operations) and are used to provide documentation and clarification of the logic, data, variables, and algorithms. Each source statement shall be self-defined or defined by a comment phrase to a level understandable by a person not associated with the original development effort. Logical groups of comment phrases may be included in a single comment statement. General comments on groups of source statements performing logical functions shall be included on separate comment statements.

5.4.5 Source record format.

5.4.5.1 Execution efficiency. Subject only to the interest of readability, clarity and maintainability, source statements shall be coded to optimize object code execution.

5.4.5.2 Indentation. Program structural indentation shall be used to improve readability and clarity.

5.4.5.3 Source statement. A source statement shall not be compound or complex in structure except as necessary to support the control structures defined in paragraph 5.3.1.

5.4.5.4 Sequence numbering. Each source record shall contain a sequence number prior to delivery as a configuration item. Sequence numbers within a procedure or routine shall be in sequentially increasing order beginning with and differing by some multiple of ten.

5.4.6 Listings. Listings related to the program shall meet the standards specified herein.

5.4.6.1 Content. For acceptance as a deliverable configuration item, the listing of a compiled program shall include source language statements and comments with resulting object machine instructions interspersed appropriately (together with actual or equivalent assembler statements, if available). Relative location of instructions and operands shall be exhibited together with statement labels, identification numbers, and card identifiers. All descriptions of referenced routines, functions, tables, variables, constants, files, indices, etc., shall be included in conjunction with this listing and arranged for convenient access.

5.4.6.2 Cross-reference listing. A cross-reference listing shall be produced relating each data name to the address of every other statement referring to it, and relating each routine and the address of other routines calling upon it. The list shall be exhibited as a sequential table in alphanumeric order.

5.4.7 Flow charts. There is no requirement that flow charts be a deliverable item.

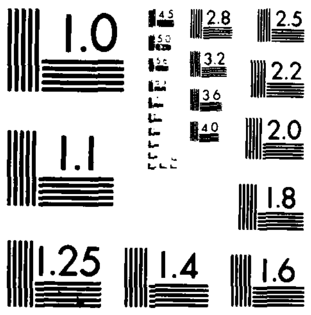
5.5 Program production. The contractor shall generate the program in an orderly and well-controlled manner. The requirements shall be translated into program design in a systematic top-down method. The system shall be divided into constituent parts and then these parts broken down into their constituents. Each level of design development (or break down) is continued until a level is reached wherein no other function is subservient to the function. Levels shall be structured so that a lower level function does not call on a higher level function. Program coding shall follow the same structure as the design, which allows identifiable division of the programming task. Programming shall commence with the highest levels which shall then be tested extensively and placed under configuration/library control before descending downward in the design to the programming of any subordinate levels. Efficient and effective control of the program during coding and test is required.

5.5.1 Organization. The contractor shall implement a program production organization that facilitates the top-down design, coding, and test of the program.

AD-A086 571 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
THE UTILIZATION OF REQUIREMENT STATEMENT METHODOLOGIES IN THE U--ETC(U)
MAR 80 F A PETRIE
UNCLASSIFIED NL

2 of 2
AD
A086571

END
DATE
FILMED
8-80
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

5.5.2 Timing and memory management. The contractor shall be responsible for management of computer system resources (e.g., main memory, mass storage, processor time, input/output controller(s), and input/output channel(s)). He shall determine the original assignment of system resources through analysis and modeling. The contractor shall monitor the utilization of the assigned resources as program development progresses. A minimum reserve of 20 percent capacity shall exist in each resource area at the time of program acceptance by the procuring agency.

5.5.3 Library usage and control. The contractor shall establish procedures for producing, updating, and controlling source and object libraries of the software under development. All initial programs and development changes shall be maintained in both source and object format. All program patches shall be maintained in maintenance/patch logs and on patch tapes until incorporated in the patch-free source program. Program patches shall, as a minimum, be identified by: patch production date, programmer producing the patch, the program segment that the patch is applicable to, the corresponding problem number or identification, the test that revealed the problem, the testing that certifies the integrity of the patch and the problem that necessitated the patch.

5.5.4 Load maps. The contractor shall describe the format, method and location in which the various portions of the program are loaded and stored in the weapon system computers and, if applicable, disks or other storage devices. This mapping shall include delineating all of the portions of the program that are to be concurrently resident in the device in question and the location and size of each portion of the program. If the system has more than one defined configuration or mode of operation for the software, the contractor shall describe this information for each configuration or mode.

5.6 Program generation.

5.6.1 Language. Weapon system software shall be coded in one of the high order programming languages (HOLs) approved by the Department of Defense unless a specific waiver has been previously granted to the procuring agency by proper authority.

5.6.2 Program regeneration. All weapon system software delivered by the contractor shall be capable of being regenerated from Government owned and the delivered support software.

5.7 Program operation. The contractor shall determine the procedures for the operation of the weapon system software. Procedures shall be described in terms understandable to operational personnel. Program operation procedures shall be subject to the approval of the procuring agency.

5.7.1 Analysis. In determining program operation procedures, the contractor shall investigate and define in detail the following areas.

5.7.1.1 Non-functional operation. Minimal processor and peripheral equipment requirements, equipment set-up for system operation, program set-up, special parameter entering requirements, standby/operate procedures, monitoring procedures, and recovery procedures shall be defined.

5.7.1.2 Functional operation. Individual operator and station functions; coordinated station procedures; all human factor aspects, modes and procedures necessary for each console or station operator to perform his function in support of system operation; the function of every control button, switch, readout and display affected by or affecting the system; all constraints imposed on operator actions shall be defined.

5.8 Quality assurance. The contractor shall implement quality assurance procedures to verify in each stage of the development that the product program will meet the current performance specifications approved by the procuring agency. The contractor shall implement quality assurance procedures to validate the accuracy, correctness and performance of the product programs, to verify the accuracy and conformance of program documentation to the requirements of this Military Standard and to ensure that all procedures incumbent on the development activity are properly and completely followed. The procedures shall be open to review by the procuring agency or its authorized representative. The implementation and functioning of the procedures shall also be open to inspection by the procuring agency or its authorized representative.

5.8.1 Organization. The quality assurance organization shall include provisions for addressing all the following facets of quality assurance.

5.8.1.1 Reporting level. The contractor's quality assurance organization shall have a corporate reporting responsibility external to the developing/engineering group to assure an objective evaluation of conformity and progress.

5.8.1.2 Participation in audits. The contractor's quality assurance organization shall present and shall conform with procedures for independent quality audits that should take place throughout the development phase starting with design development and ending with test, certification, delivery and acceptance which measure system conformance with technical and management requirements and standards.

5.8.1.3 Design reviews. The contractor's quality assurance organization shall participate in design reviews and walk throughs utilizing procedures to assure completeness and accuracy of presented materials and to assure timely and correct completion of action assignments.

5.8.1.4 Tests. The contractor's quality assurance organization shall witness tests to assure conformance with approved procedures. Quality assurance activities shall include record-keeping, maintenance, control of test materials, and conflict/discrepancy resolution.

5.8.1.5 Deliverable items. The contractor's quality assurance organization shall provide and shall conform to procedures to assure contractual correctness of all deliverable items.

5.8.1.6 Reporting. The contractor's quality assurance organization shall utilize both interdepartmental and intradepartmental reporting chains to assure prompt reporting of the results of quality assurance activities. Quality assurance shall follow-up any noted discrepancy/action assignment to assure timely and complete correction of the problem.

5.8.1.7 Authority. When conflict exists between quality assurance and other contractor functions at a specific task/management level, the conflict shall be resolved successively at the next higher level.

5.8.2 Program design. The detailed performance requirements for the weapon system software shall be audited and verified as being able to satisfy the requirements of operational requirements, operational standards and system performance specifications, as may be provided by the procuring agency.

As early as possible in the design phase, the proposed program architecture shall be verified as to its capability to support the computational load imposed by maximum operation of all functions required to be simultaneously serviced. This verification may require extensive modeling and simulation and shall, in all cases, be completed prior to design implementation and coding.

The detailed design of the weapon system software shall be verified against the performance requirements specified by the procuring agency.

The detailed performance requirements, the program architecture and the detailed program design will be subject to review by the procuring agency at scheduled milestones in the program development cycle. Prior to submission of the detailed design to the procuring agency for review, a design walk-through shall be conducted. This design walk-through shall be accomplished by one or more technically qualified persons in conjunction with the originator or originators of the detailed design.

5.8.3 Program production. Programming conventions, program design rules and programming standards shall be promulgated to and followed by all levels of program production personnel. The contractor shall insure programmers are skilled in the use of the specified language and compiler capabilities. Standard procedures shall be developed for programmers to follow in use of coding forms, submission of compile requests, reports of progress and associated listings.

A code walk-through review of each program segment shall be conducted prior to submission of the program for compile. This review shall be conducted by one or more technically qualified persons in conjunction with the originator of the code being reviewed. Coding shall be verified for complete compliance with detailed program design. Coding shall be validated for compliance with specified programming conventions and standards. Listings for developmental segments of the program shall be thoroughly desk-checked before computer-run testing.

5.9 Program test. The contractor shall determine the scope of tests required to ensure that the program being developed meets all specified technical and operational performance requirements and the acceptance criteria. The contractor shall be responsible for accomplishing all development testing. Test planning shall include development of:

a. Program acceptance criteria.
b. Levels of testing to verify performance.
c. Internal procedures for scheduling and conducting tests.
d. Detailed procedures for testing at each level.
e. Reporting procedures of test results.

All test plans, specifications and procedures shall be subject to review and approval by the procuring agency. The procuring agency shall be kept advised of all test schedules and shall be permitted to witness all tests with designated Government or contractor representatives. The contractor shall provide all supporting software necessary to conduct, control and record tests. The contractor shall define any special support software necessary to satisfactorily test the software being developed. The contractor shall identify to the procuring agency any GFE or GFI required to support the test program early enough to allow the procuring agency to obtain and deliver any such requirements without impacting the development and testing schedule.

The contractor shall provide or insure the availability of adequate facilities for conducting all required tests. The procuring agency shall have the option of specifying the facility that should be used to conduct any portion of the test program.

The contractor shall prepare test reports showing quantitative results of all tests. Such reports shall be signed by a representative of the contractor. Any formal or informal approval of the testing results by the procuring agency representative during the course of software production shall not be construed as a guarantee of the acceptance of the finished product. Testing shall consist of the following:

a. Subprogram/module tests
b. Function tests
c. System performance tests
d. Systems integration tests

5.9.1 Subprogram module tests. Each subprogram/module shall be subjected to developmental testing. Such tests shall be adequate to determine compliance with the applicable technical, operational, and performance specifications. As a minimum, each subprogram/module shall pass the following tests:

a. Verification of the coded subprogram/module to ensure that it fully satisfies the performance and design specification requirements and that all code to be delivered has been exercised.
b. Error-free compile/assembly of the coded subprogram/module.
c. Exercise of the subprogram/module in terms of input/output performance with the results satisfying the applicable performance and design specification requirements.

5.9.2 Function tests. Subprograms/modules shall have passed the subprogram/module tests prior to being subjected to functional testing. The subprogram/modules shall be integrated individually into particular subsystem programs. Function tests shall be adequate to determine compliance with the applicable technical, operational, and performance specifications.

5.9.3 System performance tests. All subsystem programs shall have passed the function tests prior to system performance testing. The subsystem programs shall be integrated individually until all subsystem programs have been integrated into the system program. These tests shall be adequate to determine compliance with the applicable technical, operational, and performance specifications. As a minimum, systems performance testing shall be performed to:

a. Verify the total man-machine interface.
b. Validate system initiation, data entries via peripheral devices, program loading, restarting, and the monitoring and controlling of system operation from display consoles and other control stations as applicable.
c. Verify system integration of equipment and subsystems.
d. Verify the capability of the system to satisfy all applicable performance and system level specification requirements.
e. Via the deliberate insertion of erroneous inputs, verify the capability of the system to properly handle and survive erroneous inputs and proper inputs entered in improper format or sequence.

5.9.4 Systems integration test. In instances where the developed program is a component of a larger system involving the integration of two or more programs developed as separate projects, the individual contractor shall be required to participate in total system integration testing. Integration testing may be conducted at facilities other than the development facility, such as a Land-Based Test Site. Each contractor shall provide technical support to the integration testing as required.

5.9.5 Software trouble reporting. The contractor shall develop and implement internal procedures for handling and reporting all software or software related problems identified. In addition to the categories and priorities described below, a code shall be utilized to indicate the status of each Software Trouble Report (STR) as it progresses through the correction cycle. All STRs shall be verified for accuracy and correctness and submitted on standard forms.

The contractor shall maintain a complete set of software problem data files throughout the duration of the contract and make this information available to the procuring agency or his authorized representative upon request.

5.9.5.1 Software trouble report category. Software problems shall be classified by category as follows:

a. Program trouble (P). The program does not operate according to supporting documentation and the documentation is correct.
b. Documentation trouble (D). The program does not operate according to supporting documentation but the program operation is correct.
c. Design trouble (E). The program operates according to supporting documentation but a design deficiency exists.
d. Logic trouble (L). The program has a logical error with no directly observable operational symptom but with the potential of creating trouble.

5.9.5.2 Software trouble report priority. Software problems shall be classified by priority as follows:

a. Priority 1 - a major malfunction rendering the entire program or a major functional area unusable or unreliable. All problems of a major nature which are unpredictable, that is, cannot be reproduced at will, shall be classified as priority 1.

b. Priority 2 - a serious malfunction which limits the program from performing its full capability and for which there is no alternative procedure available.

c. Priority 3 - a malfunction which presents an erroneous result but for which the program provides an alternative permitting full capability operation.

d. Priority 4 - a minor error or operator annoyance which has no effect on the operational capability of the system.

e. Priority 5 - an insignificant error or no error.

5.10 Program acceptance.

NOTE: This section is presented here to show the intended subject matter. The content of section 5.10 will be modified, as necessary, to be in conformance with the final version of TADSTAND X (Software Quality Assurance Testing Criteria).

Incrementally during development and prior to acceptance by the procuring agency, the contractor shall demonstrate the complete capabilities of the program. This demonstration shall take the form of meeting the incremental program performance criteria by formal testing and auditing.

Program performance criteria shall be measured by: the number of existing patch words, the priority and number of outstanding and unresolved Software Trouble Reports (STRs), the endurance run time without system failure, the core memory requirements, and the timing requirements of the operating program. These criteria shall be met incrementally, during development, prior to operational employment and throughout the software life cycle. The specific criteria and their relative times of compliance are specified in Figure 3.

These performance criteria are binding on all software program types specified in this standard.

PARAMETER	MODULE TEST	FUNCTION TEST	PROGRAM ACCEPTANCE TEST (PAT)	SYSTEM INTEGRATION TEST (SIT)
PATCH CRITERIA (NO. OF ALLOWABLE PATCH WORDS)	PATCH FREE	DELIVER PATCH FREE FOR PAT.	NO MODULE HAS A NUMBER OF PATCH WORDS GREATER THAN 2% OF MODULE SIZE. OVERALL PATCH WORDS NOT TO EXCEED 1% OF PROGRAM SIZE. NOTE: IF TIME PERMITS, THE PATCHES SHOULD BE COMPILED IN AND RETESTED PRIOR TO SIT.	NO MODULE HAS A NUMBER OF PATCH WORDS GREATER THAN 2% OF MODULE SIZE. OVERALL PATCH WORDS NOT TO EXCEED 1% OF PROGRAM SIZE.
ALLOWABLE PROGRAM ERRORS (STRs) BY PRIORITY	<u>PRIORITY</u> <u>ERRORS</u> 1 0 2 1 3 5 4 5	<u>PRIORITY</u> <u>ERRORS</u> 1 0 2 5 3 10 4 10	<u>PRIORITY</u> <u>ERRORS</u> 1 0 2 3 3 10 4 10	<u>PRIORITY</u> <u>ERRORS</u> 1 0 2 0 3 5 4 10
ENDURANCE RUN	N/A	12 HOURS	24 HOURS	36 HOURS
CORE	AS PER ALLOCATION	AS PER ALLOCATION	AS AUTHORIZED	AS AUTHORIZED
TIMING	PER DESIGN	PER DESIGN	AS AUTHORIZED	AS AUTHORIZED
COMPILER ERRORS AND WARNINGS	NONE ALLOWED	NONE ALLOWED	NONE ALLOWED	NONE ALLOWED

FIGURE 3. Program performance criteria.

a. Formal Qualification Testing (FQT) shall demonstrate the applicable performance requirements. The test environment shall consist of actual operational and interfacing equipment to the extent practicable. Data inputs shall be operational scenarios designed to demonstrate the correct response of the computer program to stimulation actions through man-machine, or other external, interfaces. The operating procedures for the program, as determined by the contractor, shall be used to exercise the program in FQT.

b. Auditing shall verify the correspondence and correlation between all deliverable items which are associated with and support each computer program entity which the contractor has been tasked to produce. This auditing shall include but is not limited to the following items:

   (1) Review of program documentation for format, completeness, correspondence and correlation.

   (2) Review program listings for compliance with applicable programming standards and conventions.

   (3) Verify operator/user manuals as complete and accurate.

   The contractor shall prepare all materials for the audit, provide space for the audit group, and provide technical assistance. The procuring agency or designated representative will direct and control the audit.

5.11 Configuration management. The contractor shall develop and implement procedures to ensure the positive identification, control, status accounting and authentication of the configuration of the weapon system software, the detailed performance requirements and the detailed program design during all phases of the development effort. The contractor shall insure that such procedures are integrated with the configuration management procedures addressing the total system when the software is only one element of the weapon system being developed. Procedures shall provide:

a. Positive identification of all program elements.

b. Rapid, comprehensive and accurate treatment of proposed changes to elements under configuration control.

c. Comprehensive implementation of approved changes and dissemination of corrected documentation and program changes.

d. Accurate records of status of all proposed changes.

e. Verifications of change control, identification and status accounting of the software products.

5.11.1 Configuration identification.

5.11.1.1 Baselines. The contractor shall establish internal baselines representing the approved description of the configuration of the weapon system software under development.

5.11.1.2 Documentation identification. The contractor shall establish titling, labeling, numbering and cataloging procedures for all descriptive documentation and program material which satisfy the following criteria:

a. Denotes the program to which it applies

b. Describes the purpose of the document

c. Defines the baseline which it is a part of, or in support of

d. Denotes the serial, edition and change status of the document

The date of program compilation shall be indicated as part of the identifier for each delivered program. Sequence numbering of a program shall be structured so future changes to the program can be properly noted.

5.11.2 Configuration control. The contractor shall establish procedures for the formal control of all documents, program materials and the development support library. Procedures shall include the establishment and functioning a software configuration control board, the methods and formats for submission and acting on Software Change Proposals, Software Enhancement Proposals, Software Trouble Reports and Specification Change Notices.

5.11.2.1 Software configuration control boards (SCCB). Each baseline shall be under the formal control of a responsible board. The board shall identify and maintain the complete and current description of each element of the baseline. The board shall consider all proposed changes to the baseline and take appropriate action on each proposal. Each proposal shall be analyzed and evaluated in the following areas:

a. Operational impact
b. Technical design impact
c. Resource requirements (e.g., cost, personnel, time)

For all approved changes, the board shall ensure implemented changes are reflected in all baseline documentation under the control of the procuring agency.

Changes which require the approval of the procuring agency shall be forwarded by the contractor with complete analysis, evaluation, and recommendations.

5.11.2.2 Software changes. MIL-STD-1679 shall be used during software development to communicate changes among the software community. Changes to the software proposed by the contractor (including descriptive documentation) which is under configuration control by the contractor or the government or both, shall be submitted to the appropriate software configuration control board(s) as either Software Change Proposals (SCP) or Software Enhancement Proposals (SEP) depending on the classification of the changes. An SCP or SEP which has cost or schedule impact shall be attached to a form DD1692 (Engineering Change Proposal, page 1) completed and numbered in accordance with reference 2.1.a.

5.11.2.3 Documentation changes. Procedures for controlling the preparation and dissemination of changes to documentation to reflect approved and implemented SCPs, SEPs, and STRs shall be developed. Such procedures shall be designed to insure the simultaneous promulgation of the documentation and program change.

5.11.3 Configuration status accounting. The contractor shall establish procedures to enable the generation of periodic status reports on all elements under configuration management. Procedures shall identify all SCPs, SEPs, and STRs in preparation, in review, and in the current stage of implementation. Procedures shall identify all disapproved and deferred SCPs, SEPs, and STRs.

5.11.4 Configuration authentication. The contractor shall utilize configuration authentication techniques which, as a minimum, include the following:

a. A review process that reconciles deliverable software products to their approved documentation.
b. Procedures to assure that the software products are identified as stated in the applicable contract requirements and the approved project configuration management plan.
c. Procedures to be used by the change control authority to confirm incorporation of the approved configuration changes.
d. Procedures for the reconciliation of configuration status accounting reports and status (version) of the software products to the approved baseline(s) and its approved changes.

5.12 Management control. The contractor shall determine and implement a management system for the development effort which is acceptable to the procuring agency. The management of the development shall emphasize efficiency and economy. Clear lines of authority and responsibility shall be established. The management system shall provide for the coordination of all facets of the development under a master schedule of events and milestones. Milestone dates shall be established for demonstrations of evolving software capabilities. Such demonstrations are intended to provide the necessary visibility for project management and meaningful output for product validation. The management system shall provide a capability to monitor the progress of the development by means of regular status reports, reviews and audits. The management system, including planning and procedural guidance for the development effort, shall be compiled in an overall plan for visibility, formalization, control, and coordination of the development.

5.12.1 Organization. The contractor may use an internal organization of his own choice, subject only to the requirements from this standard which are invoked by the procuring agency. The contractor shall designate an overall manager for the development effort. The functions of design, production and test shall be given organizational visibility. The relationship of all support functions, both full-time and part-time, required to support the development effort shall be clearly defined. The responsibilities of all sub-contractors, if used, shall be clearly visible to the procuring agency.

5.12.2 Resource management. The contractor shall determine his resource requirements in the three areas of personnel, facilities, and equipment. Planning shall be completed early enough to permit orderly acquisition, installation and training (if applicable), of resources on an optimum schedule to prevent delay and to avoid dead-time. Planning shall be responsive to schedule changes. The contractor shall avoid sharp fluctuations in personnel requirements by judicious shifting of personnel as development tasks change.

Reusability, permanency or length of project and convenience of location shall be weighed. The procuring agency may direct the use of government or other facilities.

The contractor shall consider the cost-effectiveness of commercial equipment to assist in the development where appropriate. Where weapon system equipment is Government-furnished or Government-specified, the contractor shall be responsible only for the cost-effectiveness of its use and maintenance, not its acquisition. The possibility of continuing use of the equipment by the Government during the operational support phase of the software life-cycle shall be a consideration. The contractor shall implement a system of management monitoring of utilization in the areas of personnel, facilities, and equipment considering both quantity and cost. Actual utilization rates shall be compared to predicted rates at least monthly. The procuring agency may specify more frequent comparison. Variations shall be expeditiously investigated and corrective action initiated. Personnel stability and productivity shall be measured regularly.

5.12.3 Status reviews. Status reviews may be requested by the procuring agency at regular intervals during the development effort. The contractor shall be able to provide information at these reviews to apprise the procuring agency of current status, progress, problems, and critical items occurring in the development effort within the purview of the contractor.

5.12.3.1 Status review subjects. The contractor shall address the following subjects, as appropriate to the stage of the development effort, in each status review:

a. Organizational changes, managerial personnel changes
b. Program design status
c. Development schedule status (milestone prognosis)
d. Coding status
e. Software Trouble Report (STR) status
f. Software Change Proposal (SCP) status
g. Software Enhancement Proposal (SEP) status
h. Integration schedule status
i. Testing status
j. Deliverables
k. Progress on previous problems
l. New action items/problems .

m. Delinquencies: governmental, outside contractor, subcontractor, and internal
n. Manpower utilization
o. Facilities utilization
p. Computer system resource utilization (see 5.5.2)
q. Financial summary

5.12.3.2 Status review subject items. Within each subject area , the contractor shall cover the following items, as applicable:

a. The program schedule updated to the end of this reporting period.
b. Major difficulties encountered and plans to overcome them, including: Tasks/units that are currently behind schedule (or have anticipated schedule changes), their effects on completion of the project, and steps being taken to remedy schedule delays.
c. Other information which defines cause and effect of significant changes on the contract schedule.
d. Problems which actually or potentially will cause deviation from contractual requirements.
e. Summary of meetings and conferences held during the reporting period including action items with due dates for both the contractor and the procuring agency. Current status of action items shall be included until reported closed.

5.12.3.3 Documentation reviews. Documents and programming materials as specified, shall be scheduled for detailed review prior to approval or acceptance. The purpose of the review shall be to:

a. Verify that the subject documents and programming materials comply completely and accurately with the performance requirements or design specifications of the previous documents and programming materials and all other standards and constraints imposed by the procuring agency.

b. Validate the accuracy and completeness of the documents and programming materials by checking for all components, their correct cross-reference and editorial accuracy.

The reviews shall be in two stages; a preliminary working-level review, followed by a formal (or critical) review after changes resulting from the preliminary review have been entered. Reviews shall be scheduled by the contractor, with the concurrence of the procuring agency, and in accordance with milestones in the software development plan. The procuring agency may designate other activities to participate in the review. The contractor shall distribute drafts of review documents and programming materials to each designated activity sufficiently in advance of the scheduled preliminary review to allow adequate internal review by each activity. The contractor shall distribute a corrected version of the review documents and programming materials after completion of the preliminary review. The critical review for the acceptance or approval of the documents and programming materials shall expeditiously follow the distribution of the corrected version.

5.12.3.4 Special reviews. Special reviews may be scheduled by the procuring agency at major milestones or events in the development effort not covered by Baseline Reviews or Status Reviews. A special review of the test program as developed shall be conducted. The contractor shall furnish the same support for special reviews as for baseline reviews.

5.12.4 Inspections and audits. The procuring agency may employ a physical inspection to determine the contractor's conformace with contractual requirements. As a minimum, areas of interest include development facilities, documentation controls, deliverable data items, Government-imposed standards, and contractor internal standards.

a. Facilities. The development and test facilities may be inspected for contractual conformity at any time during the life of a software system development contract.

b. Configuration management. Contractor conformance with the approved Configuration Management Plan may be audited through examination of records and attendance at change control Board meetings.

c. Internal standards. The procuring agency may audit the contractor's conformance with internal standards of software development and control.

d. Quality assurance. The procuring agency may audit and inspect the contractor's conformance with the approved Software Quality Assurance Plan.

60.1 Section 1, Scope. The content of Section 1 of a computer program development specification shall be as defined in the following example:

Example:

1. SCOPE

1.1 Identification. This paragraph shall contain the approved identification, nomenclature, and authorized abbreviation for the computer program.

1.2 Functional summary. This paragraph shall contain a brief description of the overall computer program by major functions (tasks). It shall further identify and summarize the specification content, composition, and intent.

60.2 Section 2, Applicable documents. The content of this Section 2 shall be in accordance with 4.2.

60.3 Section 3, Requirements. This is the major section of the computer program development specification. It shall consist of a series of paragraphs that specify in detail the performance requirements of the computer program. This section shall define and specify all functional performance requirements, design constraints, and standards necessary to ensure proper development of the computer program. This paragraph shall contain a brief general discussion of the overall system within which the program will operate. It shall show the relationships of each subsystem with the computer program portion of the system. In particular, the role assigned to the computer program should be stressed to delineate the functions it must accomplish for the system. As the introductory segment of the specification, this paragraph shall:

a. Provide a brief general discussion of the overall system and make reference to other system or subsystem performance specifications that will further clarify the performance requirements of the subject system.

b. Provide a general description of the peripheral equipment with which the specified program will interface.

c. Provide a general description of any programs with which the specified program will interface.

d. Provide a general description of the major functions of the computer program relative to the manner in which they will be subsequently treated.

60.3.1 *Paragraph 3.1, Program definition.* This paragraph shall provide a detailed description of the major functions of the computer program. This paragraph shall:

a. Detail the requirements imposed on the computer program by each interfacing equipment and shall include purpose of equipment, computer interface description, equipment options and controls, and timing and accuracy limitations.

b. Provide timing and sequencing interface requirements imposed by other computer programs or by equipment or operational limitations.

c. Describe the major functions of the computer program including their interaction, sequencing and timing requirements. Block diagrams of the interfaces shall be provided to facilitate presentation of the material.

60.3.2 *Paragraph 3.2, Detailed functional requirements.* The subparagraphs under this paragraph shall contain the necessary detailed text and mathematical descriptions for each of the required computer program functions. A set of subparagraphs shall be prepared for each major function or subfunction, whichever is required for clarity. Descriptive and introductory material for each function shall be included as necessary in this paragraph.

60.3.2.1 *Paragraph 3.2.1, Inputs.* This paragraph shall provide a detailed description of all input data. Source of the input, method of insertion, and validity checks shall be defined. Quantity and timing of the

*(Omit if not applicable)

input data and associated limits shall be specified. Operator control requirements shall be detailed, including names and descriptions of operator actions, consoles or operator positions where applicable, and the required programmed restrictions.

60.3.2.2 *Paragraph 3.2.2, Processing.* This paragraph shall provide a textual and mathematical description of each of the processing requirements of each function. Presentation of the mathematical descriptions under each function shall include:

a. Purpose – This area shall describe the exact intent of the mathematical operation(s). This involves a definition of the specific input and output parameters and the processing required.

b. Approach – This area shall contain a textual description of each mathematical operation specified. The accompanying narrative shall identify accuracies required, sequence and timing of events, and relevant restrictions or limitations. Derived equations shall be shown with appropriate mathematical and control symbols adequately defined.

c. Diagrams of Geometry – Suitable diagrams shall be included in the text produced under the preceding paragraphs where applicable.

60.3.2.3 *Paragraph 3.2.3, Outputs.* This paragraph shall provide a detailed description of all output data, control parameters, and displays. Method and timing of outputs shall be described completely. Operator output requirements (e.g., hard copy, CRT displays) must include name, content, timing, format and routing of the information.

60.3.2.4 *Paragraph 3.2.4, Special requirements.* This paragraph shall contain detailed descriptions of special data processing requirements or instructions for special formats to accommodate testing, recording, simulation, necessary procedures, system growth requirements, recovery requirements, and special personnel requirements.

60.3.3 *Paragraph 3.3, Adaptation.* These paragraphs shall contain a description of the data requirements with respect to system environment, system parameters, and system capacities. Adaptation data is that data that can be centrally modified as needed to define the scope of operational functions within prescribed limits. These data are divided into three classes and presented as follows.

60.3.3.1 *Paragraph 3.3.1, General environment.* This paragraph shall contain a description of environmental data detailing the characteristics anticipated for all particular installations. Each installation will select and set the required data and value for operational use. Examples of such data are: grid limits, radar ranges and areas of coverage, prescribed safety limits, etc.

60.3.3.2 *Paragraph 3.3.2, System parameters.* This paragraph shall contain a description of constants required by one or more subprograms that may change from time to time incrementally within a specified range according to operational needs. Such data consists of allowable trajectory deviations, missile performance characteristics, etc.

60.3.3.3 *Paragraph 3.3.3, System capacities.* This paragraph shall contain a description of the capacity requirements for the computer program. Items such as compatibility for total simultaneous target handling, total number of simultaneous missile trajectory controls, total number of simultaneous displays and operator station requests, etc., shall be described. The system capacities are directly related to computer storage capacities, interfacing subsystem timing rates, and interfacing equipment capacities.

60.4 **Section 4, Quality assurance provisions.** This section shall specify test/verification requirements, methods of verification, and the necessary test tools and facilities to conduct the required tests/verifications. This section shall establish the requirements for the test plans and procedures that must be formulated for verification of the program. The intent of the test effort is to verify that the performance requirements as stated in Section 3, of the specification have been met. The following paragraphs shall be included.

60.4.1 *Paragraph 4.1, Introduction.* This paragraph shall establish the requirement for development of a test plan and test procedures for the subject program. It shall specify the following levels of testing:

a. Computer subprogram testing

b. Computer program testing

c. Computer program acceptance testing

d. System integration testing

60.4.2 *Paragraph 4.2, Test requirements.* This paragraph shall specify the requirements for each level of testing except the acceptance test level. For each level, the test tools and facilities required shall be specified. The requirements shall include test formulas, algorithms, techniques and acceptable tolerance limits, as applicable.

60.4.3 *Paragraph 4.3, Acceptance test requirements.* This paragraph shall establish the means by which the procuring agency may formally accept the computer program as fulfilling the performance requirements.

NOTE: Since the depth of coverage possible in this section depends upon the type of program to be tested, *the minimum* essential content of this section shall include the establishment of the levels of tests required and the requirement for production of test plan and test procedures documents.

60.5 Section 5, Preparation for delivery. This section is normally not applicable.

60.6 Section 6, Notes. This section shall include information that is stated for administrative convenience only, and is not a part of the specification in the contractual sense, e.g., it shall not include requirements that constrain design or development, or qualify the performance requirements. This section shall include a list of all documents, specifications, etc., that are necessary for program development and that are not included with this specification.

60.7 Section 10, Appendix I. This section of the specification shall contain requirements which are contractually a part of the specification but which, for convenience in specification maintenance, are incorporated herein, e.g., requirements of a temporary nature or for limited effectivity. Appendixes may be bound as separate documents for convenience in handling, e.g., when only a few parameters of the program are classified, an appendix containing only the classified material may be established. Where parameters are placed in an appendix, the paragraph of Section 10 shall be referenced in the main body of the program specification in the place where the parameter would normally have been specified. Typical data that may be included in computer program development specification appendixes include:

a. Mathematical derivations

b. Alternate method

c. Summary of equations

d. Definitions of terms

LIST OF REFERENCES

[1] Davis, C.G. and Vick, C.R., "The Software Development System," IEEE Transactions on Software Engineering, pp. 69-84, January 1977.

[2] Willis, R.R. and Jensen, E.P., "Computer Aided Design of Software Systems," 4th International Conference on Software Engineering, pp. 116-125, 1979.

[3] Mullery, G.P., "CORE--A Method for Controlled Requirement Specification," 4th International Conference on Software Engineering, pp. 126-135, 1979.

[4] Hammond, L.S., Murphy, D.L. and Smith, M.K., "A System for Analysis and Verification of Software Design," COMPSAC 78, pp. 42-47, November 1978.

[5] Munson, J.B., "Software Maintainability: A Practical Concern for Life Cycle Costs," COMPSAC 78, p. 54, November 1978.

[6] Balzer, R. and Goldman, N., "Principles of Good Software Specification and Their Implications for Specifications Languages," 1979 IEEE Specifcations of Reliable Software, pp. 58-67, September 1979.

[7] Heninger, K., "Specifying Software Requirements for Complex Systems: New Techniques and Their Application," 1979 IEEE Specifications of Reliable Software, pp. 1-14, September 1979.

[8] Merten, A. and Teichreow, "The Impact of Problem Statement Languages on Evaluating and Improving Software Performance," AFIPS Conference Proceedings, pp. 849-857, 1972.

[9] Jones, C., "A Survey of Programming Design and Specifications Techniques," 1979 IEEE Specifications of Reliable Software, pp. 91-103, September 1979.

[10] Teichreow, D., "A Survey of Languages for Stating Requirements for Computer-Based Information Systems," AFIPS Conference Proceedings, pp. 1203-1224, 1972.

[11] TRW Defense and Space Systems Group, Software Requirements Engineering Methodology, SREP Final Report - Volume 1., August 1977.

[12] Bell, T., Bixler, D. and Dyer, M., "An Extendable Approach to Computer-Aided Software Requirements Engineering," IEEE Transactions on Software Engineering, pp. 49-59, January 1977.

[13] Alford, M., "Software Requirements Engineering Methodology (SREM) at the Age of Two," COMPSAC 78, pp. 332-339, November 1978.

[14] Alford, M., "A Requirements Engineering Methodology for Real-Time Processing Requirements," TRW Software Series, September 1976.

[15] Teichreow, D. and Hershey, E., "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Transactions on Software Engineering, pp. 41-48, January 1977.

[16] Krohn, M. and Williamson, R., "Towards an Automatic System Generator," Software 72, pp. 72-76, 1972.

[17] Furia, N., "A Comparative Evaluation of RSL/REVS and PSL/PSA Applied to Digital Flight Control System," AIAA Conference Proceedings, pp. 330-337, December 1979.

[18] Heninger, K. and others, Software Requirements for the A-7E Aircraft, Naval Research Laboratory, Washington, D.C. 20375, Memorandum Report 3876, 27 November 1978.

[19] Office of Management and Budget, Circular No. A-109, 5 April 1976.

[20] Department of Defense Directive 5000.1, Major Systems Acquisition, 18 January 1977.

[21] Department of Defense Directive 5000.2, Major System Acquisition Process, 18 January 1977.

[22] Department of Defense Directive 5000.29, Management of Computer Resources in Major Defense Systems, 26 April 1976.

[23] Department of Defense Instruction 5010.21, Configuration Management Implementation Guidance, 6 August 1968.

[24] Department of Defense Military Standard 1679 (NAVY), Weapon System Software Development, 1 December 1978.

[25] Secretary of the Navy Instruction 3560.1, Navy Tactical Digital Systems Documentation Standards, 8 August 1974.

[26] Department of Defense Military Standard 490, Specification Practices, 30 October 1968.

[27] De Roze, B. and Nyman, T., "The Software Life Cycle -- A Management and Technological Challenge in the Department of Defense," IEEE Transactions on Software Engineering, pp. 309-318, July 1978.

[28] Wolverton, R., "Software Life Cycle Management -- Dynamics Practice," Second Software Life Cycle Management Workshop, Atlanta, Georgia, 21-28 August 1978.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Professor N.F. Schneidewind, Code 52Ss Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	LT Frederic A. Petrie III, USN 217 Riverside Drive Morganton, North Carolina 28655	2

END
DATE
FILMED
8-80
DTIC